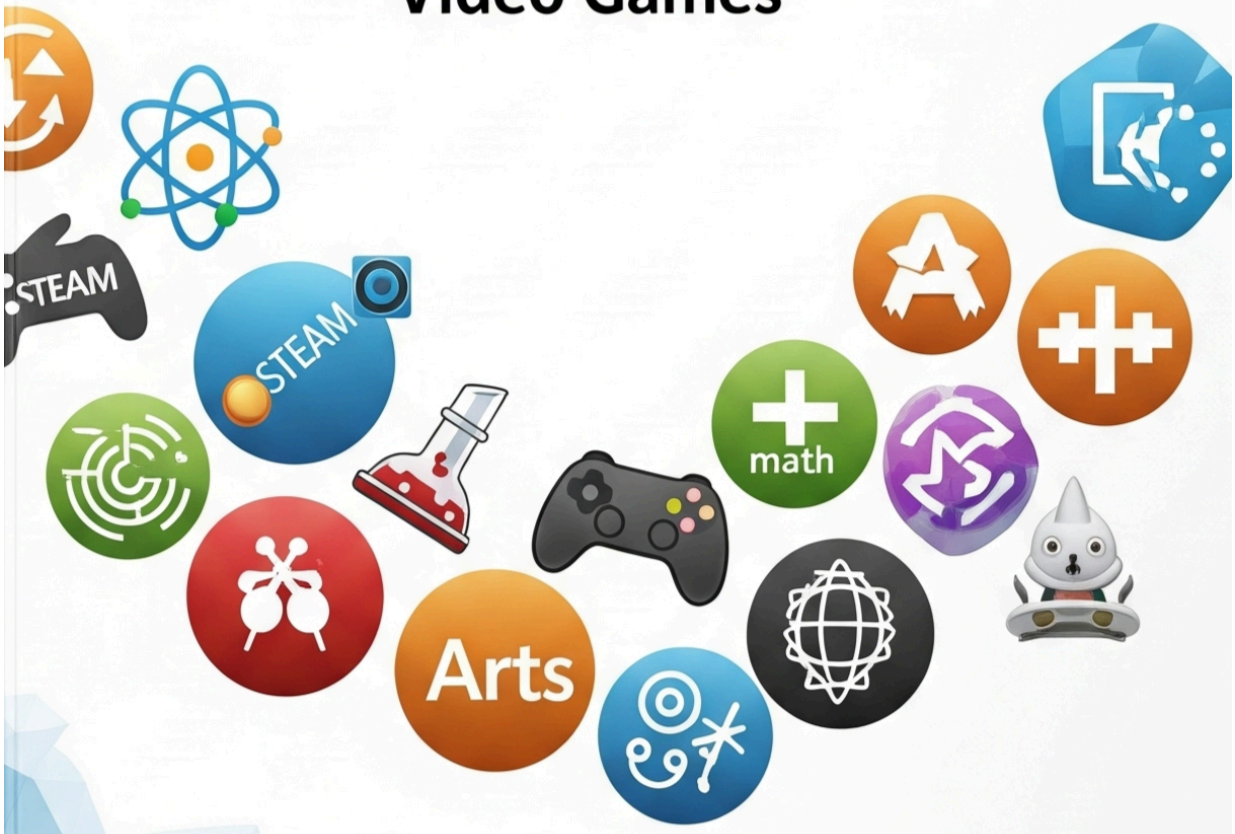
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ
ΙΚΥ

Raising Interest and Excellence in STEAM Education

Incorprated with Non-Formal Based Video Games



Project No: 2024-1-EL01-KA2220-SCH-000250956

Table of Contents

INTRODUCTION

In the 21st century, the world changes rapidly due to technological developments, globalization, and new social and economic challenges. Education systems must keep up with these changes to prepare students for an unpredictable future. One of the key areas that supports this preparation is STEAM education—Science, Technology, Engineering, Arts, and Mathematics. STEAM helps learners develop critical thinking, problem-solving, creativity, collaboration, and innovation. However, many education systems still face serious challenges in delivering STEAM education in a way that motivates students and connects with real-life skills.

One of the biggest problems is that many teachers are not fully prepared to teach STEAM subjects using modern and innovative approaches. In many classrooms, traditional teaching methods dominate. These methods rely mostly on textbooks, lectures, and exams. They often do not encourage students to explore, create, or apply knowledge in real-world situations. As a result, students lose interest in STEAM subjects, and they do not develop the necessary skills to succeed in modern life and future careers.

This project aims to solve these problems by creating a strong partnership between countries to develop and implement new and creative lesson scenarios in STEAM education. These lesson scenarios include non-formal learning methods, especially through the use of video games. Non-formal education gives students the chance to learn outside traditional classroom structures. It focuses more on experience, creativity, interaction, and active participation. Video games, in particular, offer a powerful tool to support learning. They allow students to try, fail, learn from their mistakes, and solve problems in engaging and dynamic environments.

The use of video games in education is not a new idea, but this project takes a step further by combining video games with STEAM subjects in a multidisciplinary way. When students learn through video games, they develop digital skills, logical thinking, and creativity. They become more motivated, and their academic performance improves. Games like Scratch, mBlock, and Minecraft are used to design and support learning scenarios. These platforms are student - friendly, easy to use, and allow students to create their own learning experiences. When teachers use these tools in STEAM lessons, students participate actively and take responsibility for their learning.

In this project, teachers from six different countries create a total of 36 lesson scenarios. Each country creates 6 scenarios that combine STEAM subjects with non-formal, video game-based learning activities. These scenarios follow a multidisciplinary approach. This means that the lesson plans connect two or more subjects from the STEAM fields, such as combining science with technology, or engineering with mathematics and art. The goal is to show students how knowledge from different areas works together in real life. For example, when students design a city in Minecraft, they use mathematical calculations, scientific principles, and creative design. This type of learning helps students understand the importance and usefulness of STEAM in everyday life.

Teachers design the lesson scenarios by including video game elements that they create using Scratch, mBlock, or Minecraft. They include interactive activities, experiments, group work, challenges, and reflection tasks. The activities encourage students to explore problems, discuss

their ideas, and build new knowledge through trial and error. Each scenario includes clear learning objectives, connections to school curriculum, assessment strategies, and opportunities for teamwork and creativity.

At the same time, the project also focuses on improving the skills of STEAM teachers. Many teachers still feel unprepared to use digital tools or teach through interdisciplinary methods. They may not know how to create or use video games in education, or they may lack experience with student-centered learning. For this reason, the project provides training and support for teachers. They learn how to design games, how to plan effective lessons, and how to guide students through project-based and game-based learning. The goal is to increase teachers' confidence and abilities so they can deliver high-quality STEAM education in their classrooms.

Through this process, teachers not only improve their individual teaching skills, but they also collaborate with colleagues from different countries. They share ideas, compare methods, and create new strategies together. This cooperation helps build strong professional networks between teachers, schools, and educational leaders. It also encourages the sharing of best practices and the development of a European community of innovative STEAM educators. These networks continue to grow even after the project ends, creating long-term benefits for the education systems involved.

The lesson scenarios that teachers develop through this project cover many different STEAM topics and connect them to real-world situations. For example, students may use Scratch to simulate the movement of planets and learn about physics and programming at the same time. In another scenario, they may use mBlock to create a game that explains the water cycle or energy transfer. With Minecraft, they may design a sustainable city, explore environmental problems, or build historical reconstructions. These examples show how video games help students understand complex ideas in a fun and engaging way.

Each lesson scenario includes built-in assessments, not only through tests but also through student reflection, group feedback, and project presentations. This supports deeper learning and helps teachers understand students' progress. It also shows students that learning is more than just passing exams—it is about building understanding, solving problems, and being creative.

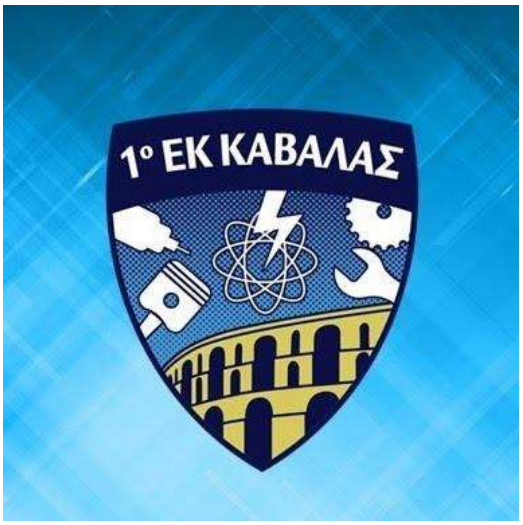
The project also supports national and international education goals. It matches the priorities of the European Education Area, the EU Digital Education Action Plan, and global frameworks like UNESCO's Education 2030 Agenda and the UN Sustainable Development Goals. It focuses on equity, inclusion, and quality in education. It promotes lifelong learning and prepares students for the digital world. The project gives special attention to students who may not be interested in traditional teaching. By using games and interactive tools, it reaches a wider group of learners and gives all students the chance to succeed in STEAM subjects.

In the long term, the project aims to create a strong, sustainable impact. It produces a collection of 36 ready-to-use, high-quality lesson scenarios that schools across Europe can adapt and use. These materials are flexible, creative, and based on the real needs of students and teachers. They also support educational policymakers by offering new models for modern and engaging STEAM education. The project's results show how to improve student performance, teacher quality, and the learning environment in a way that is practical, scalable, and future-oriented.

Students who participate in these lessons not only improve their academic skills, but they also gain important life skills. They learn how to work in teams, solve problems, manage projects, and think critically. They become more confident, more motivated, and more curious about the world. These are the exact qualities that students need to face the challenges of the future and to become active, responsible citizens in a fast-changing world.

This project understands that teachers play a central role in education. They are not only knowledge providers, but also facilitators, designers, and mentors. When teachers have the tools, training, and support they need, they can create powerful learning experiences for their students. That is why this project invests in teacher development as much as it does in student learning.

In conclusion, the project brings together schools, teachers, and education experts from across Europe to create and implement a new model of STEAM education. By integrating non-formal, video game-based learning with interdisciplinary lesson scenarios, it transforms the teaching and learning of STEAM subjects. It raises student interest, supports teacher innovation, and improves educational quality. Most importantly, it prepares students for a world where knowledge, creativity, and digital skills are key to success. Through cooperation, creativity, and commitment, this project creates lasting change in how we teach and learn STEAM across Europe and beyond.



1st EK KAVALAS Learning Scenarios

Learning Scenario 1: Crisis Response – Designing an Emergency Aid Station	
Target Groups:	Student§s aged 14–18 (upper secondary or vocational) in: • Health sciences • Emergency care and nursing • Applied STEAM fields • Civil protection or rescue services Teams of 2–3 students.
Learning Objectives:	By the end of the session, students will: • Understand spatial and operational planning in emergency aid. • Apply basic triage and crisis management strategies. • Simulate emergency scenarios using Minecraft's collaborative environment. • Use Redstone or Code Builder for automated alerts and simulated systems. • Demonstrate leadership, teamwork, and communication in crisis conditions.
Materials & Resources:	• Minecraft Education Edition with student accounts • Printed triage cards and scenario briefings • Code Builder enabled on all devices • Projector for teacher demonstrations • Worksheets for patient logs and planning • Optional: Minecraft camera/portfolio for screenshots
Introduction (15 minutes):	1. Present a simulated **train crash** scenario: *"100 people injured. You're the first medical team on site. You must set up and run an emergency aid station."* 2. Introduce key triage principles (START method, color codes: Red, Yellow, Green, Black). 3. Divide students into teams and distribute printed role cards representing injury levels. 4. Show an example Minecraft triage setup and explain the team's responsibilities.
Scenario Presentation (Step-by-Step)	
Students may use CodeBlocks (optional) to auto-place lighting or build ramps. *Challenge 1: Design the Emergency Triage Station (20 minutes)* • Build a clear zone layout: • **Red Zone:** Critical condition • **Yellow Zone:** Serious, not critical • **Green Zone:** Walking wounded • **Black Zone:** Deceased • Add labeled tents, signage, and paths for stretcher/wheelchair access. • Use NPCs or signs to represent patients and symptoms. *Challenge 2: Activate Emergency Alert Systems (15 minutes)* • Build **Redstone or CodeBlock systems**: • Chat command "emergency" triggers flashing Redstone lamps. • Create **alert buttons** near each zone using pressure plates or levers. • (Optional) Simulate power failure or aftershock to test team reactions. *Challenge 3: Real-Time Triage & Adaptation (15 minutes)* • Halfway through, introduce **a second wave of injuries** or a simulated **fire or collapse** near the Red Zone. • Teams must: • Reprioritize existing patients • Relocate tents or create safety barriers • Make decisions under time pressure • Document decisions and actions	
Teacher Tips and Strategies:	• Encourage verbal simulation: "Call for backup," "Apply pressure to wound," etc. • Ask reflective questions: *"What made this patient Red?", "Where would you move the unconscious child?"*

	• Emphasize function and clarity over visual perfection. • Walk around to monitor teamwork and decision-making logic.
Learning Outcomes and Skills:	• Emergency planning & triage knowledge • Communication and collaboration under stress • Spatial reasoning and problem-solving • Redstone logic or basic coding with Code Builder • Adaptability and decision-making in real-time scenarios
Feedback Collection (10 minutes):	Use either discussion or a quick reflection form: • What was your biggest challenge? • What would you change in your station layout? • How did your team handle the crisis adaptation?
Assessment & Evaluation:	Rubric-based evaluation on: • Clarity and functionality of the triage station • Realistic application of triage principles • Team collaboration and responsiveness to scenario changes • Presentation of patient management strategies • Use of Redstone or Code Builder to simulate alerts (Optional: Upload screenshots to a shared class portfolio)
Conclusion (5 minutes):	Summarize key lessons on triage, teamwork, and space design in emergency care. Emphasize the connection between gaming, critical thinking, and real-life health sciences. Encourage students to reflect on careers in emergency response, nursing, or engineering for resilience planning.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1_hirdoWQviNV2Ay1_Y2I9_pRDu8dUiKv5gQMCDcx8mo/edit?usp=sharing
Game Link:	Recommended: Start with a **flat world** or custom terrain with marked plots for each team. https://education.minecraft.net/en-us

Learning Scenario 2: Disaster Triage Game – Designing and Managing an Emergency Aid Station	
Target Groups:	Ages 14–18 (Upper Secondary/VET) Ideal for health science, emergency response, or vocational nursing students Group size: 2–3 students (small collaborative teams)
Learning Objectives:	By the end of this session, students will be able to: • Understand the basic principles of disaster triage and emergency site planning. • Apply real-time decision-making and prioritization through a simulated Minecraft triage station. • Collaborate effectively to build and organize emergency response spaces. • Demonstrate critical thinking and communication in crisis situations. • Reflect on the practical challenges of emergency aid deployment.
Materials & Resources:	• Minecraft Education Edition installed on all devices • Student logins and access to Code Builder • Printed Triage Role Cards (Red/Yellow/Green/Black - injury levels) • Scenario Handouts: “Train Crash – 100 Casualties” • Timer or countdown app • Optional: projector to show group builds • Challenge Sheet (provided separately) • Link to CodeBlocks for Redstone triage automation (provided in separate file)
Introduction (15	Introduce the scenario: "A major train crash has occurred. You are first responders building and operating an Emergency Aid Station in a remote location."

minutes):	Discuss the basics of disaster triage (Start/JumpSTART method, color codes, priorities). Assign students to teams and distribute the Triage Role Cards (indicating types of patients they must simulate responding to). Demonstrate a basic triage layout in Minecraft — tents, marked zones, signage. Quick walk-through on how to open Code Builder (press "C") and basic command block structure (if applicable).
Scenario Presentation (Step-by-Step)	
Mission Briefing (5 minutes): Explain the mission: Teams must build a functioning triage zone in Minecraft and manage 10 coded patient cases based on realistic symptoms. Each team is responsible for shelter, assessment, and priority assignment. **Challenge Launch (60 minutes):** Students are tasked with: 1. Designing the layout of the emergency triage area (clearly labeled zones: Red, Yellow, Green, Black). 2. Building treatment tents or shelters for each category. 3. Using signs or NPCs to document symptoms, simulate patients. 4. Optionally, using Code Builder/Redstone to simulate real-time alerts (e.g., arrival of more injured, changing conditions). 5. Prioritizing care and reporting status during periodic check-ins. **Patient Simulation (20 minutes):** At the 30-minute mark, introduce a second wave of injuries or a sudden event (e.g., a fire near the station or loss of power). Students must adapt their build, reprioritize cases, and collaborate under pressure. **Presentation (15 minutes):** Each team presents a brief tour of their triage station, explaining layout choices, how they classified injuries, and how they responded to sudden developments.	
Teacher Tips and Strategies:	• Encourage verbal role-play to simulate medical teamwork. • Reinforce time pressure to mimic emergency scenarios. • Circulate and ask open-ended questions: "Why did you place this patient in Yellow?", "What would you do if this case deteriorates?" • Emphasize clarity, organization, and realistic design over aesthetic perfection.
Learning Outcomes and Skills:	• Emergency planning and patient prioritization • Collaboration and leadership under pressure • Realistic application of health and safety principles • Basic coding or Redstone usage for alerts and automation • Spatial reasoning in high-stakes environments
Feedback Collection (10 minutes):	Hold a short group debrief or reflection survey: • What was the most difficult decision your team made? • How did your team collaborate under stress? • What would you change in a real-world triage center?
Assessment & Evaluation:	• Participation and engagement during simulation (rubric-based) • Completeness and functionality of the triage layout • Correct application of triage principles (based on their patient logs) • Quality of reflection and crisis response adaptation
Conclusion (5 minutes):	Summarize key takeaways: • How space, structure, and speed matter in emergency response • The importance of prioritization and clarity under stress • Real-world relevance of triage and emergency aid stations
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1ZyFHCTcIF_Y5hXcgDUaolgzoq3zouyYi04x8pdwkV1c/edit?usp=sharing

Game Link:	https://education.minecraft.net/en-us

Learning Scenario 3: Elderly Care Community – Designing an Accessible Living Space	
Target Groups:	Upper secondary or vocational students (ages 14–18) in the fields of healthcare, social care, architecture, or applied STEAM programs.
Learning Objectives:	• Understand the daily living needs of elderly individuals. • Design spaces with accessibility, comfort, and safety in mind. • Apply spatial reasoning and universal design principles in Minecraft. • Foster empathy through user-centered design. • Collaborate in teams to build purposeful environments.
Materials & Resources:	• Minecraft Education Edition (student accounts required) • Computers with Code Builder enabled • Projector or screen for teacher walkthrough • Printable planning worksheets • Real-life references (images or floorplans of assisted living spaces) • Optional: Minecraft camera/portfolio tool for student documentation
Introduction (15 minutes):	Begin with a discussion on aging and the challenges elderly individuals face in daily life. Use real-world examples to highlight barriers such as stairs, narrow halls, poor lighting, or unsafe bathrooms. Present the task: Design a small accessible living community in Minecraft that meets the needs of seniors. Assign students to small teams and introduce the three core challenges they must complete.
Scenario Presentation (Step-by-Step)	
Challenge 1: Design a Safe & Accessible Living Unit (35 minutes) **Goal:** Build a 1-bedroom senior-friendly home with: • No stairs or steps • Wide doors and hallways • Proper lighting • Handrails or guide blocks in key areas • Clear signage and furniture placement Students may use CodeBlocks (optional) to auto-place lighting or build ramps. *Challenge 2: Community Features for Mobility and Comfort (30 minutes)* **Goal:** Create shared areas that support physical, emotional, and social well-being, including: • A garden path or exercise area with smooth surfaces • A central gathering space (community kitchen, dining hall, or library) • A clinic or first aid center with beds and supplies Use gentle slopes (not stairs), benches, signs, and open spaces to show accessibility. --- *Challenge 3: Emergency Response Readiness (30 minutes)* **Goal:** Include features that help elderly residents during emergencies: • Emergency call buttons (Redstone buttons or pressure plates) • Flashing alert lights (Redstone lamps) • Easy exits with directional signage • A team roleplay of a safety drill (verbal or text command simulation) Optional CodeBlock: Alert lights with on chat command "alert" to activate Redstone.	
Teacher Tips and Strategies:	• Encourage empathy by asking “Would a person using a wheelchair access this area?” • Guide teams to observe real-life designs (hall rails, ramps, automatic doors). • Prompt iterative thinking: test, revise, and adjust designs. • Use peer walkthroughs to promote critical design reflection.
Learning Outcomes and Skills:	• Awareness of accessibility and aging-in-place design • Critical thinking in spatial planning • Creative use of Minecraft tools for simulation

	• Collaboration and role-based teamwork • Introductory Redstone/CodeBuilder logic for automation
Feedback Collection (10 minutes):	Students complete a reflection form or group discussion with prompts: • What design feature are you most proud of? • What was the hardest accessibility problem you encountered? • How did your team approach safety and comfort?
Assessment & Evaluation:	• Completion and quality of the three challenge builds • Realism and thoughtfulness of accessible features • Group communication and planning • Presentation and reflection responses • Optional: Screenshots or Portfolio submissions
Conclusion (5 minutes):	Recap the value of designing with empathy. Emphasize how Minecraft can simulate real-world design thinking in healthcare and housing. Encourage students to reflect on how their learning could apply to nursing, architecture, occupational therapy, or elder care.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/19s73Iue_bv2G5wLDjabsrpdgPNiTQoPbLgK6jSBWZZg/edit?usp=sharing
Game Link:	Recommended: Start with a **flat world** or custom terrain with marked plots for each team. https://education.minecraft.net/en-us

Learning Scenario 4: Maternal Health Unit – Designing a Prenatal and Postnatal Care Center	
Target Groups:	Upper secondary or vocational students (ages 14–18) in the fields of health sciences, midwifery, childcare, nursing, or applied STEAM programs.
Learning Objectives:	• Understand the spatial and functional needs of prenatal and postnatal care units • Design healthcare environments that prioritize comfort, hygiene, and emotional safety • Apply real-world medical principles in Minecraft Education Edition • Foster empathy for maternal and infant care through user-focused design • Collaborate in small teams to build supportive medical environments
Materials & Resources:	• Minecraft Education Edition with student accounts • Code Builder enabled on all devices • Printable planning sheets and design checklists • Real-world references (photos or blueprints of maternity wards) • Optional: Minecraft camera or portfolio documentation tool
Introduction (15 minutes):	Begin with a class discussion on the importance of maternal health, highlighting challenges during prenatal, childbirth, and postnatal periods. Show real-world images of birthing centers, recovery rooms, and mother-infant bonding spaces. Explain the task: Build a Prenatal and Postnatal Care Center in Minecraft that supports mothers, newborns, and medical teams. Divide students into teams of 2–3 and outline the three core challenges.
Scenario Presentation (Step-by-Step)	
Challenge 1: Prenatal Check-Up Zone (35 minutes) **Goal:** Design a welcoming and functional space for expectant mothers to receive regular care. **Requirements:** • Waiting lounge with comfortable seating and lighting • Check-up room with examination beds, desks, and medical supplies • Signage and NPCs with wellness advice or pregnancy tips **Optional CodeBlock:** /prenatalZone to place beds, signs, and lighting	

Challenge 2: Birthing and Recovery Area (30 minutes)
Goal: Create a secure and supportive room for labor, delivery, and postnatal recovery.
Requirements:

- At least one delivery room with privacy (dividers, curtains, enclosed space)
- Postnatal area with beds for mothers and bassinets for newborns
- Sanitation features (wash stations, sinks)
- Gentle lighting and décor for a soothing environment

Optional CodeBlock: /birthRoom to auto-place bed, divider, and newborn area

---

Challenge 3: Family Bonding and Infant Care Room (30 minutes)
Goal: Build a space where new families can relax and learn about infant care.
Requirements:

- Comfortable area with couches, toys, and bookshelves
- Feeding and diaper changing station
- Signs or NPCs offering care tips, breastfeeding info, or parenting advice
- Calm, home-like atmosphere

Optional CodeBlock: /familyCorner to place couch, bookshelves, and decorative items

Teacher Tips and Strategies:	• Ask: "How does this room support the mother's comfort and dignity?" • Remind students to think about both medical functionality and emotional well-being • Use walkthroughs to guide students in improving spatial layout and flow • Highlight real-world roles: midwives, nurses, lactation consultants
Learning Outcomes and Skills:	• Knowledge of maternal healthcare environments • Empathy and user-focused design principles • Application of STEAM tools in healthcare simulation • Team communication and construction planning • Basic coding with CodeBuilder for interior automation
Feedback Collection (10 minutes):	Use a quick reflection worksheet or group circle: • What aspect of your design best supported mothers and babies? • What would you add or improve in a real clinic? • How did your team divide roles and make decisions?
Assessment & Evaluation:	• Completion and quality of all three challenge zones • Realism and empathy in space design • Team communication and presentation • Use of Minecraft and CodeBuilder features • Optional: Student screenshots or narrated Minecraft tour
Conclusion (5 minutes):	Recap the significance of maternal and newborn care spaces. Encourage students to reflect on careers in nursing, midwifery, architecture, or public health. Emphasize how digital tools can model real-world healthcare solutions.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1DbZOz4AwnVqdhtVj5ZfleekYDnMHTA10Sk54csD31F4/edit?usp=sharing
Game Link:	https://education.minecraft.net/en-us

Learning Scenario 5: Mental Health Support Centre – Creating a Safe Space	
Target Groups:	Upper secondary or vocational students (ages 14–18) in the fields of social care, health sciences, or applied STEAM.
Learning Objectives:	• Understand key components of post-pandemic recovery in community settings. • Design supportive, inclusive spaces that address physical and emotional needs. • Apply principles of resilience, public health, and environmental planning in Minecraft. • Promote teamwork and empathy through community-focused design. • Integrate CodeBuilder and Redstone tools for simulation and automation.
Materials & Resources:	• Minecraft Education Edition (student accounts required) • Computers with Code Builder enabled • Projector or screen for teacher walkthrough • Printable worksheets and sketches of safe spaces • Optional: Minecraft camera/portfolio tool for screenshots
Introduction (15 minutes):	Begin with a discussion on how communities recover after a pandemic. Explore the importance of safe quarantine areas, access to food and water, community spaces, and mental health support. Present the task: Design a **Recovery Camp** that supports a fictional village after a pandemic. Assign students to small teams and introduce the three main challenges.
Scenario Presentation (Step-by-Step)	
Challenge 1: Safe Entry and Welcome Area (35 minutes) Goal: Design an inviting and non-intimidating space for entry and reception. • The space should include: • Reception area with warm lighting, benches, and plants • Signs or NPCs with calming messages • Paths made of natural blocks leading to main areas Optional CodeBlock: /welcomePath – Creates a stone slab path and places flowers along it. Challenge 2: Reflection and Therapy Rooms (30 minutes) Goal: Create rooms that allow for quiet thinking, privacy, and emotional support. • The space should include: • 1–2 private therapy rooms with beds, bookshelves, and decor • Quiet zone with windows, plants, and paintings for reflection • Emotion station: signs or books to share feelings or messages Optional CodeBlock: /therapyRoom – Builds a chair, bookshelf, and flower pot. Challenge 3: Group Healing and Activity Space (30 minutes) Goal: Design shared spaces that support social connection and creative expression. • The space should include: • Mindfulness or art therapy garden with flowers and seating • Open room with musical blocks or creative stations • Positive affirmations displayed throughout the area Optional CodeBlock: /groupGarden – Builds grassy area with repeated flower planting.	
Teacher Tips and Strategies:	• Ask reflective questions such as: 'Would someone feel calm and welcome here?' • Use real-world therapy rooms or sensory spaces as examples. • Allow students to personalize their areas with detail blocks. • Encourage feedback between teams during mid-session walkthroughs.

Learning Outcomes and Skills:	• Empathy-driven spatial design • Understanding of mental health support structures • Use of Minecraft tools for simulation • Teamwork and idea exchange • Creative thinking for user-centered experiences
Feedback Collection (10 minutes):	Students complete a reflection form or discuss in groups. Suggested prompts: - What made your space feel emotionally safe? - How did your team choose which features to prioritize? - What would you do differently in a real-life support center?
Assessment & Evaluation:	• Completion and quality of the three space designs • Alignment with mental health support principles • Creativity and realism in layout • Presentation and group reflection • Optional: Upload screenshots and share Minecraft portfolios
Conclusion (5 minutes):	Recap the importance of emotional safety and mental health in design. Encourage students to reflect on how they might apply what they've learned in their schools, communities, or future careers.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1DbZOz4AwnVqdhtVj5ZfleekYDnMHTA10Sk54csD31F4/edit?usp=sharing
Game Link:	https://education.minecraft.net/en-us

Learning Scenario 6: Post-Pandemic Village Relief – Building a Recovery Camp	
Target Groups:	Upper secondary or vocational students (ages 14–18) in the fields of healthcare, community planning, social care, applied STEAM, or civil protection.
Learning Objectives:	• Understand key components of post-pandemic recovery in community settings. • Design supportive, inclusive spaces that address physical and emotional needs. • Apply principles of resilience, public health, and environmental planning in Minecraft. • Promote teamwork and empathy through community-focused design. • Integrate CodeBuilder and Redstone tools for simulation and automation.
Materials & Resources:	• Minecraft Education Edition (student accounts required) • Computers with Code Builder enabled • Projector or screen for teacher walkthrough • Printable scenario worksheets and village layout sketch • Real-life references (images of mobile clinics, community kitchens, etc.) • Optional: Minecraft camera/portfolio tool for student documentation
Introduction (15 minutes):	Begin with a discussion on how communities recover after a pandemic. Explore the importance of safe quarantine areas, access to food and water, community spaces, and mental health support. Present the task: Design a **Recovery Camp** that supports a fictional village after a pandemic. Assign students to small teams and introduce the three main challenges.
Scenario Presentation (Step-by-Step)	
Challenge 1: Health & Safety Zone (35 minutes) **Goal:** Design a structured area for quarantine, hygiene, and recovery. The zone must include: • An **isolation tent** with 2–4 beds, clearly separated from main areas • A **handwashing and sanitation station** (with cauldrons or water tanks) • A **wellness tent** for light exercise and mental recovery • Signs and fencing to separate entry and exit points	

Optional CodeBlock: "/buildIso" to auto-place isolation beds and walls.

Challenge 2: Community Regeneration & Resilience (30 minutes)
Goal: Create shared areas that support long-term well-being and cohesion.
Suggested elements:

- A **community garden** for shared food production
- An **open-air classroom** or digital hub for student learning
- A **story circle or announcement board** for emotional expression and information
- Open walkways, signs, and seating areas to encourage safe movement and interaction

Optional CodeBlock: "/gardenPath" to place grass paths and flowers for garden layout.

Challenge 3: Future Emergency Preparedness (30 minutes)
Goal: Prepare the village for future crises with proactive infrastructure.
This zone should include:

- A **supply depot** (with labeled chests: food, medicine, PPE)
- An **alert system** using Redstone or CodeBuilder
- **Health information stations** using NPCs or signs with messages
- A **roleplay drill** for what to do in case of a new outbreak or emergency

Optional CodeBlock: "/alertNow" to flash Redstone lamps simulating an emergency alert.

Teacher Tips and Strategies:	• Encourage reflection: "Would this support people physically and emotionally?" • Refer to real-world examples (mobile clinics, modular shelters, etc.) • Guide teams to sketch designs before building • Promote feedback and idea-sharing between groups mid-build
Learning Outcomes and Skills:	• Knowledge of community recovery principles and wellness infrastructure • Application of design thinking in health and environmental planning • Teamwork and collaborative decision-making • Use of Minecraft and CodeBuilder for realistic simulations • Emotional intelligence and empathy for real-world crises
Feedback Collection (10 minutes):	Use group discussion or reflection forms with prompts such as: • What was the most important feature in your camp? • What would be your team's first response if a new virus emerged? • How did your design address both physical and mental health?
Assessment & Evaluation:	• Completion and quality of the three design challenges • Relevance and realism of the features created • Group collaboration and innovation • Explanation and presentation of recovery strategies • Optional: Upload screenshots and submit Minecraft portfolio
Conclusion (5 minutes):	Recap how design thinking can help communities rebuild after major health crises. Encourage students to think about how their learning could apply in real-life health, planning, or STEAM-related careers.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1pRvuH2onaLzA3ZKekCkg0rBZAERvH9hlO_dw8H9v5vA/edit?usp=sharing
Game Link:	Recommended: Start with a **flat world** or custom terrain with marked plots for each team. https://education.minecraft.net/en-us

Learning Scenarios University of Cukurova

Learning Scenario 1: Eco-Engineers Survival: Coding to Save Our World	
Target Groups:	• Age: 14–18 years old • Level: Middle to High School (Beginner to Intermediate coding experience) • Group Size: Individual players or small teams (2–3 students)
Learning Objectives:	By the end of the lesson, students will: • Understand basic coding concepts (loops, conditionals, random numbers) using Minecraft Code Builder. • Apply scientific principles like sustainability, renewable energy, and ecosystem restoration. • Develop problem-solving and critical thinking skills in a simulated environment. • Foster creativity and artistic expression through pixel art design. • Collaborate and communicate effectively in team-based tasks. • Reflect on mathematical patterns like prime numbers in creative building.
Materials & Resources:	• Computers with Minecraft Education Edition installed • Student accounts/logins • Access to Code Builder (press "C" key in-game) • Projector/Screen (optional, for group demonstration) • Printable Challenge Sheet (the PDF we created) • Pen & paper for planning • Timer or stopwatch
Introduction (15 minutes):	• Briefly discuss STEAM and its importance. • Introduce the idea: "The world needs eco-engineers!" • Show an example video or screenshots of Minecraft coding (optional). • Quickly explain how to access and use Code Builder. • Divide students into teams if working collaboratively.
Scenario Presentation (Step-by-Step)	
1. **Mission Briefing (5 minutes):** Explain their mission — restore the barren world, build energy farms, defend villages, and inspire through art. 2. **Hand out the Challenge Sheet (2 minutes):** Review all 6 Challenges. 3. **Walkthrough Code Blocks (8 minutes):** Quickly show starter examples (tree planting, solar panel grid). 4. **Launch the Game (5 minutes):** Students spawn into the custom/flat Minecraft world.	

5. **Challenge Time (60 minutes):** Students tackle the 6 tasks using coding blocks. 6. **Bonus Extension (optional):** Offer extra coding challenges for faster teams.	
Teacher Tips and Strategies:	• Model coding slowly once — for example, plant 1 tree manually first, then show how loops automate it. • Roam the room to answer coding syntax questions without solving everything for them. • Encourage testing often: remind students to test code frequently rather than waiting until the end. • Support collaboration: allow teammates to split work (one codes irrigation, another builds solar, etc.). • Celebrate errors: Coding bugs are part of the journey!
Learning Outcomes and Skills:	• Communication — working in teams and explaining code. • Problem-Solving — fixing bugs and optimizing code. • Critical Thinking — planning solutions before coding. • Creativity — designing pixel art and layout strategies. • Math Skills — understanding grids, coordinates, and number patterns. • Science Knowledge — learning about renewable resources.
Feedback Collection (10 minutes):	• Short survey OR group discussion: What was the hardest challenge? What was the most fun part? How would you improve your code if you had more time?
Assessment & Evaluation:	• Participation and Engagement: Active contribution during coding sessions (rubric: Excellent / Good / Needs Support). • Completion of Challenges: How many tasks did they complete (scale: 0–6)? • Code Review: Check their use of loops, conditionals, and creative modifications. • Reflection Quality: Students share learnings and challenges during feedback.
Conclusion (5 minutes):	• Celebrate success! • Highlight the real-world connections between coding and environmental science. • Remind them: Coding can create real-world change — not just in Minecraft, but in future careers too.

Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1rNnVsvgJ9hmZgRKw3G8z1Llof3TZzeVSEcXE7o6sHsk/edit?tab=t.0
Game Link:	https://education.minecraft.net/joinworld/MywxNiw4LDE2

Learning Scenario 2: "Elemental Quest: A Minecraft Adventure in Science"	
Target Groups:	• Age: 14-18 years old • Subject: Science (Physics, Chemistry, Biology, Earth Science) • Level: High School students, 9th-12th grade • Prerequisites: Basic understanding of Minecraft gameplay; introductory knowledge of basic science concepts (gravity, chemical reactions, ecosystems, weather).
Learning Objectives:	**By the end of this lesson, students will be able to:** 1. Apply scientific principles (such as physics, chemistry, biology, and earth science) through coding commands in Minecraft. 2. Use command blocks in Minecraft to simulate real-world scientific concepts (e.g., gravity, chemical reactions, animal behavior, weather control). 3. Design and execute a series of challenges using Minecraft's Redstone and command blocks to demonstrate scientific principles. 4. Collaborate with peers to complete challenges in a team-based environment, integrating their knowledge of science and coding.
Materials & Resources:	**Materials & Resources:** • Minecraft Education Edition or Java Edition with access to command blocks. • Computers with Minecraft installed. • Internet Access for researching specific commands or science concepts (optional). • Projector or whiteboard for explaining the lesson and showing key concepts. • Command Blocks for each student or group, with pre-configured codes for challenges. • Redstone and related materials for activating the command blocks. • Printed or Digital Worksheet for tracking progress and reflecting on the scientific principles.

Introduction (15 minutes):	**Introduction (10-15 minutes):** **1. Teacher's Introduction:** Begin by introducing the concept of using Minecraft as a tool for learning about science. Explain that today's lesson will involve completing different challenges related to Physics, Chemistry, Biology, and Earth Science within the Minecraft world using command blocks. Briefly review key concepts that students will explore: how gravity works, how chemical reactions happen, how ecosystems function, and how weather can be controlled. **2. Motivation:** Show a short video or give a brief demo of how Minecraft's command blocks work, especially how players can simulate real-world science scenarios. Give examples: "Today, we'll make plants grow faster, make gravity disappear, control the weather, and simulate chemical reactions—all within Minecraft."

Scenario Presentation (Step-by-Step)

1. **Overview of the Stations (5-10 minutes):**

   Present the four science stations that students will be working through:

   - **Physics Station:** Simulate gravity and force effects.
   - **Chemistry Station:** Create chemical reactions.
   - **Biology Station:** Simulate ecosystems, plant growth, and animal behavior.
   - **Earth Science Station:** Control the weather and climate.

   Explain that each station has a command block system that they will interact with to trigger specific effects.

2. **Instructions (5 minutes per station):**

   Physics Station: Students will use the /effect command to give themselves levitation, simulating gravity manipulation. Encourage them to experiment with the command's parameters.

   Chemistry Station: Students will simulate a chemical reaction, such as creating fire when interacting with wood, using the /execute and /summon commands.

   Biology Station: Students will spawn animals like cows or trees using commands based on biome and environmental conditions.

   Earth Science Station: Students will control weather patterns using commands like /weather thunder or /time set day.

3. **Overview of the Stations (5-10 minutes):**

   Present the four science stations that students will be working through:

   - **Physics Station:** Simulate gravity and force effects.

- **Chemistry Station:** Create chemical reactions.
- **Biology Station:** Simulate ecosystems, plant growth, and animal behavior.
- **Earth Science Station:** Control the weather and climate.

Explain that each station has a command block system that they will interact with to trigger specific effects.

4. **Instructions (5 minutes per station):**

Physics Station: Students will use the /effect command to give themselves levitation, simulating gravity manipulation. Encourage them to experiment with the command's parameters.

Chemistry Station: Students will simulate a chemical reaction, such as creating fire when interacting with wood, using the /execute and /summon commands.

Biology Station: Students will spawn animals like cows or trees using commands based on biome and environmental conditions.

Earth Science Station: Students will control weather patterns using commands like /weather thunder or /time set day.

5. **Team-Based Execution (10 minutes per station):**

Divide students into small teams (2-4 students per team).

Assign each team a station and have them follow the instructions to complete their science challenges.

Students will explore the use of Redstone to trigger different commands, experiment with the effects, and work together to complete each challenge.

Teacher Tips and Strategies:

1. **Explain Redstone Basics:**

Ensure students understand how Redstone works as it can be used to activate command blocks.

Demonstrate how pressure plates or buttons can trigger the commands.

2. **Provide Support for Coding:**

Some students may be new to Minecraft command blocks. Provide step-by-step guidance for writing and debugging simple commands.

Walk around to assist with troubleshooting errors and encourage students to test their systems before moving to the next challenge.

3. **Encourage Experimentation:**

Suggest that students experiment with the parameters in their commands. For example, change the duration of the levitation effect or spawn different animals in their ecosystem.

Ask them to predict the outcome of certain changes before running the command.

4. **Challenge Extensions:**

   If students finish early, challenge them to combine the scientific principles from multiple stations into a single integrated system. For example, using gravity to trigger a chemical reaction followed by animal spawning in a specific biome.

Learning Outcomes and Skills:

Students will:

1. Gain practical knowledge of command blocks in Minecraft and how they relate to real-world scientific concepts.
2. Develop skills in collaborative problem-solving through team-based activities.
3. Learn how to apply coding principles to simulate real-world phenomena in a game environment.
4. Enhance their scientific understanding of topics such as gravity, chemical reactions, ecosystems, and weather control.
5. Improve their critical thinking and creativity as they troubleshoot and design their own command block systems.

Feedback Collection (10 minutes):	1. **Reflection**: After completing the stations, have students reflect on what they learned about the scientific concepts. Ask them to write a brief summary of their experience, highlighting what worked well and what they found challenging. 2. **Peer Feedback**: Allow students to share their solutions with other groups, explaining how they approached the challenges. Encourage constructive feedback between groups to foster learning.

Assessment & Evaluation:	**1. Assessment Criteria:** Evaluate students based on their creativity and accuracy in applying scientific principles through their command block systems. Assess their ability to collaborate effectively in teams and troubleshoot their code. Review their reflections and explanations for clarity and understanding of the scientific concepts they implemented. **2. Formal Assessment (Optional):** Provide a short quiz at the end of the session to test understanding of the scientific principles covered (e.g., "How does gravity work in Minecraft?" or "What happens when you combine fire and wood in Minecraft?").
Conclusion (5 minutes):	**1. Wrap-Up:** Recap the science concepts students explored in Minecraft (gravity, chemical reactions, ecosystems, weather). Encourage students to think about how they might apply their new knowledge in the real world or in other games. **2. End Reflection:** Ask students to share what they learned from the activity and how they might use Minecraft as a tool for future learning.

Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1Oos1pjbmg2ySeUZqDt1sRclSYS2kpw_gMciOgbHeVMk/edit?tab=t.0
Game Link:	https://makecode.com/_f1DVWo4z9EpA

Learning Scenario 3: "Math Quest: The Code of the Blocks" – Solving Mathematical Challenges	
Target Groups:	• Age Group: 14-18 years • Subject: Mathematics (STEAM Education) • Skill Level: Intermediate to Advanced (students should have basic knowledge of algebra, geometry, and programming concepts) • Technology: Minecraft Education Edition, Code Builder feature

Learning Objectives:	By the end of the lesson, students will: 1. Apply mathematical concepts (geometry, algebra, probability, volume, and surface area) in a practical, interactive way using Minecraft. 2. Use basic coding commands to manipulate the Minecraft world. 3. Enhance problem-solving skills through engaging, game-based activities. 4. Develop teamwork, communication, and logical thinking through collaborative code-based challenges.
Materials & Resources:	1. Minecraft Education Edition (installed on computers or tablets) 2. Code Builder (available within Minecraft Education Edition) 3. Computers/Tablets with internet access for coding and accessing Minecraft 4. Pre-designed Minecraft world with math-based puzzles (provided by the teacher) 5. Projector (for whole-class demonstrations) 6. Printed handouts or digital resources (optional: math problems related to the lesson)
Introduction (15 minutes):	• Greeting and Ice-breaker: Welcome the students and engage them with a short discussion about Minecraft. Ask: "How many of you have used Minecraft before? How do you think we can use Minecraft to solve math problems?" • Context Setting: Explain that today's lesson combines math concepts with coding in Minecraft. Students will complete math-based puzzles in a game format while learning how to use programming to manipulate the world around them. • Learning Goal Introduction: Highlight the learning objectives and explain how these math concepts are used in real-life scenarios like architecture, game development, and engineering.

Scenario Presentation (Step-by-Step)	
1. Level 1: Geometry Gates **Description:** Students will be tasked with building geometric shapes (e.g., squares, rectangles) and calculating their area and perimeter. **Task:** Code the building of a 5x5 square and calculate the perimeter. **Teacher Action:** Demonstrate how to use the /fill command to build a shape. Discuss the formula for perimeter. **2. Level 2: Algebraic Bridge** **Description:** To cross the bridge, students will solve linear equations and place the correct number of blocks to unlock the next stage. **Task:** Solve equations like 2x + 3 = 13 and use the solution to place the corresponding number of blocks. **Teacher Action:** Guide students on how to solve for x and then input the solution into Minecraft to place the required blocks. **3. Level 3: Probability Pathway** **Description:** Students will simulate random events, such as a coin flip or dice roll, and calculate the probability of certain outcomes. **Task:** Code a random event generator and calculate the probability of rolling an even number. **Teacher Action:** Show how to use the Math.random() function for randomness and how to track results. **4. Level 4: 3D Maze of Volume and Surface Area** **Description:** Students will encounter 3D objects, such as cubes and rectangular prisms, and must calculate their volume and surface area. **Task:** Build a rectangular prism with given dimensions and calculate its volume and surface area. **Teacher Action:** Demonstrate how to build 3D shapes and calculate their volume and surface area using formulas. **5. Level 5: Final Equation Challenge** **Description:** This level combines everything learned—students must solve equations, build shapes, calculate volumes, and simulate random events to complete the challenge. **Task:** Complete a series of interconnected tasks (find perimeter, solve equations, build a cube, roll dice) to unlock the final reward. **Teacher Action:** Monitor progress, offer hints, and encourage problem-solving strategies.	
Teacher Tips and Strategies:	1. **Encourage Collaboration:** Divide students into small groups so they can work together, share ideas, and troubleshoot. Minecraft fosters teamwork, and collaborative coding can enhance learning.

	2. **Use the Projector for Demonstrations:** Show step-by-step instructions on the projector for coding and solving problems. Students can then replicate these steps on their own devices. 3. **Adapt Challenges for Skill Level:** Adjust the complexity of the challenges based on the skill level of the students. For example, in Level 2, provide simpler equations or allow more time for completion. 4. **Provide Hints When Stuck:** If students are struggling, offer guiding questions. For instance: "What mathematical formula can we use to calculate the perimeter?" or "What happens when we multiply the length, width, and height for volume?" 5. **Positive Reinforcement:** Celebrate progress with verbal recognition. Offer praise when students successfully build shapes, solve equations, or simulate random events.
Learning Outcomes and Skills:	By the end of the lesson, students should be able to: 1. Understand and apply mathematical concepts (geometry, algebra, probability, volume, and surface area) in a hands-on, interactive environment. 2. Write simple code using Minecraft's Code Builder to manipulate the Minecraft world. 3. Use logical thinking to solve mathematical problems and troubleshoot code. 4. Work collaboratively in teams to complete challenges and share knowledge. 5. Strengthen problem-solving and computational thinking through real-world applications of math.
Feedback Collection (10 minutes):	• **Exit Ticket:** At the end of the lesson, ask students to submit an exit ticket (either physical or digital) with responses to the following questions: What math concept did you enjoy solving the most? How did the coding help you understand the math better? What was challenging about today's lesson? What would you like to explore next in Minecraft-based learning? • **Class Discussion:** After the activity, hold a brief class discussion where students can share their experiences and give feedback on the challenges.

Assessment & Evaluation:	1. **Formative Assessment:** Observe students as they work through each level. Evaluate their coding ability, problem-solving approach, and understanding of mathematical concepts. 2. **Peer Review:** Have students evaluate their peers' solutions to challenges. For example, after solving an equation, they could check each other's work for accuracy. 3. **Final Evaluation:** At the end of the lesson, assess the students' ability to solve the final combined challenge. This will test their understanding of the entire lesson.
Conclusion (5 minutes):	• **Reflection:** Summarize the lesson by asking students to reflect on the experience. "What did you learn today that you can apply outside of the classroom?" • **Reinforce Key Concepts:** Briefly review the math concepts covered—geometry, algebra, probability, volume, and surface area. • **Preview Next Lesson:** Provide a teaser for the next lesson. For example, "Next time, we'll explore how to use coding to create our own 3D models and solve real-world engineering problems." • **Final Praise:** Congratulate students for completing the challenges and solving the puzzles using coding and math!
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1StA9nGdkS64Mn7Gdg-W-8tLmbEVBiBPYyt-httClYIU/edit?tab=t.0#heading=h.jc2f44j4bsbd
Game Link:	https://makecode.com/_8w7eDo6vKVuV

Learning Scenario 4: Pixel Monument Build-Off: Creating Digital Art with Minecraft Code Blocks	
Target Groups:	• Students aged 14–18 • Middle and High School Art, Computer Science, or STEAM classes
Learning Objectives:	• Understand how coding can be used to create artistic designs. • Develop algorithmic thinking by using loops, arrays, and conditions. • Apply creative problem-solving to complete a digital building task. • Collaborate and communicate effectively in a digital space.
Materials & Resources:	• Minecraft Education Edition or Minecraft with Code Builder enabled • Computers or tablets (one per student or pair) • Internet access (optional for tutorials) • Graph paper and colored pencils • Teacher pre-built code (provided) • Projector (optional for demonstration)
Introduction (15 minutes):	1. Start with a discussion: **"Can coding be an art form?"** 2. Show examples of **pixel art** and **Minecraft pixel sculptures**. 3. Briefly introduce the day's challenge: "You'll use Minecraft's code blocks to automatically build your own pixel art monument based on a creative theme!"
Scenario Presentation (Step-by-Step)	
Step 1: Brainstorm (5 min) • Students randomly choose or are assigned a theme (e.g., Dreams, Nature, Mythical Creatures). **Step 2: Pixel Art Design (10 min)** • Students sketch a 10x10 grid design on paper using colored pencils. • Each color represents a different block. **Step 3: Code Setup (5 min)** • Open Minecraft, press C, and open Code Builder with Block coding. • Create the pixelArray matching their design. **Step 4: Coding (15 min)** • Students insert provided codes: Set up the pixelArray Create the buildArt function • Modify blocks in code to match their own colors. **Step 5: Building and Testing (10 min)**	

- Run /buildArt to see their art automatically build.
- Debug if needed (wrong block placed, wrong position).

Optional Challenge:

- Advanced students can add /animateArt to animate a second version!

Step 6: Showcase (5 min)

- Students share and explain their final pixel monuments.

Teacher Tips and Strategies:	• Pair weaker coders with stronger ones for peer support. • Encourage creativity — remind them that pixel art doesn't have to be "perfect." • Walk around and assist with debugging loops or fixing agent errors. • Show how small changes in arrays = big visual changes.
Learning Outcomes and Skills:	• **Coding Skills:** Understanding loops, lists, conditions. • **Artistic Skills:** Pixel art design, color theory application. • **Problem-Solving:** Debugging code, logical structure thinking. • **Collaboration:** Pair programming, peer teaching. • **Creativity:** Unique interpretation of abstract themes.
Feedback Collection (10 minutes):	• Quick exit ticket: Students answer 2 questions on a sticky note or online form: "What part of coding the art was hardest?" "If you could add anything to your pixel art, what would it be?"
Assessment & Evaluation:	**Category** / **Criteria** / **Points** Creativity — Unique design based on theme — 10 pts Coding Quality — Proper use of loops and conditions — 10 pts Technical Execution — Successfully builds with no major bugs — 10 pts Presentation — Explains their process clearly — 5 pts Collaboration (if in pairs) — Effective teamwork and sharing — 5 pts *(Total: 40 pts)*
Conclusion (5 minutes):	• Recap key points: "Today you learned how loops, arrays, and block coding can bring your artistic visions to life in Minecraft." • Encourage students to continue exploring how coding can create not just games, but art and innovation.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1eza8J-fjKLZ5qC56KNl04HhremweepMFehKcawr8ons/edit?tab=t.0

Game Link:	https://makecode.com/_RF9bR2iki8gf

Learning Scenario 5: Engineering and Maths: The Tower Survival Challenge	
Target Groups:	• Students aged 14–18 years • Secondary school or high school • Classes in Mathematics, Engineering Basics, Technology/STEAM
Learning Objectives:	By the end of the lesson, students will be able to: • Apply engineering principles such as structure stability and base-to-height ratio. • Solve mathematical problems involving area, volume, and proportions. • Use block-based coding to automate challenges inside Minecraft. • Collaborate and think critically under time and environmental pressure.
Materials & Resources:	• Devices with Minecraft Education Edition (or Bedrock + Code Builder) • Internet connection (optional for multiplayer) • Code Builder (MakeCode) installed/accessible • Lesson Handout (rules, scoring, important maths formulas) • Whiteboard or Smartboard for brainstorming • Timer (for challenge countdown)
Introduction (15 minutes):	1. **Hook:** Show a quick 1-minute video of famous tower failures (e.g., Tacoma Narrows Bridge, Leaning Tower of Pisa). 2. **Discuss:** *"What makes a tower strong?"* 3. **Goal Announcement:** *"Today, you will become Minecraft Engineers. You must build a tower that can survive earthquakes and windstorms—using coding, maths, and strategy!"*
Scenario Presentation (Step-by-Step) **Step 1: Launch the Challenge** • Students type /start to begin. • Instructions are displayed in-game. **Step 2: Initial Building Phase** • Students have 10 minutes to build a basic tower (max 20 blocks tall).	

Step 3: Unlock Better Materials • Students type /materials to answer maths quizzes. • Correct answers earn stone, cobblestone, concrete.	
Step 4: Testing Phase • After 20 minutes of building, students run /wind to simulate storms. • Then, /quake to simulate an earthquake. **Step 5: Survival Check** • Students type /check. • Towers taller than 20 blocks and still standing are winners!	
Teacher Tips and Strategies:	• Group Students into teams of 2–3 for collaboration. • Use a visible timer to create urgency (e.g., "5 minutes left for building!"). • Prompt strategic thinking: *"Should you build tall first or strengthen the base?"* • Differentiate quizzes: Offer harder maths problems for bonus materials. • Allow one rebuild round if they fail after the first quake!
Learning Outcomes and Skills:	**Students will develop:** • **Maths Competency:** Applying area, surface area, proportions. • **Engineering Mindset:** Problem-solving under real-world physics simulations. • **Coding Literacy:** Understanding event-driven programming (chat commands, conditions). • **Collaboration:** Working in teams under time pressure. • **Critical Thinking:** Adjusting designs based on environmental feedback.
Feedback Collection (10 minutes):	• **Quick exit ticket questions:** What was the biggest engineering challenge you faced? How did math help you in building a better tower? What coding block did you find most useful? (Optional: Use an online form like Google Forms, or a simple sticky-note board.)
• Limited materials (wood, dirt, sand).	

Assessment & Evaluation:	**Formative Assessment:** • Observe teamwork and strategy discussions. • Check if students apply proportional rules and maths correctly. **Summative Assessment:**

- Tower survives both tests and meets height requirements (20 blocks or higher).
- Successful completion of at least 2 coding quizzes.

Optional grading rubric out of 100 points:

- 30 points: Tower design and engineering.
- 30 points: Correct maths answers.
- 20 points: Code usage and automation.

20 points: Team collaboration and problem-solving.

Conclusion (5 minutes):	• Recap what they learned about structures, physics, and coding. • Celebrate successful engineers with a "Minecraft Engineering License" certificate. • Invite students to reflect: *"If you could rebuild, what would you change?"*
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1ZcmsoDb7PNOT8zNDAm9o9IzMXB1pMEvjYCMTBiOollw/edit?tab=t.0
Game Link:	https://makecode.com/_i0HRHb7MkUkq

Learning Scenario 6: Eco-Engineers: Build & Power the City with Minecraft Code Blocks	
Target Groups:	• Students aged 14–18 years • STEAM elective classes, coding clubs, art-tech crossover workshops • Suitable for beginner to intermediate Minecraft coders
Learning Objectives:	By the end of this lesson, students will be able to: • Use coding blocks to program the Minecraft agent for resource gathering and building. • Design and engineer eco-friendly structures using Minecraft materials. • Integrate art and design into functional structures. • Apply resource management strategies. • Collaborate effectively in small teams for complex tasks.
Materials & Resources:	• Devices with Minecraft Education Edition or Minecraft Bedrock (with Code Builder access) • Access to MakeCode in block-based mode • Pre-created flat world map (base area and resource zones) • Lesson slides (optional) • Timer or countdown app (for challenges) • Scoring sheet (optional, for challenge points)
Introduction (15 minutes):	• Quick discussion: "What makes a city eco-friendly?" • Show real-world examples: solar panels, green parks, energy-saving designs. • Explain: Today you'll become eco-engineers and artists, coding your agent to build a sustainable city!
Scenario Presentation (Step-by-Step)	
Step 1: Set Up World (5 min) • Teacher loads the flat world with building supplies. • Students teleport to the starting base. **Step 2: Code Gathering Agent (10 min)** • Students open Code Builder. • Teacher shares the Resource Gathering code block (gather command). • Students test it by collecting resources. **Step 3: Code Building Structures (15 min)** • Share Auto-Build House code block (buildhouse command). • Students build their first house near the base.	

Step 4: Renewable Energy Setup (10 min) • Introduce the Solar Panel code (solar command). • Students place renewable energy devices. **Step 5: Artistic Park Creation (15 min)** • Share the Park Building code (park command). • Students design small parks, add artistic touches manually. **Step 6: Team Eco-City Challenge (20 min)** • In pairs or small groups: Plan the city layout (homes, parks, power) Use code creatively to finish their city before time runs out!	
Teacher Tips and Strategies:	• Walk around and coach teams on optimizing code (loops, inventory use). • Encourage custom commands (e.g., "buildtower", "makefountain"). • Emphasize design and efficiency: better cities, not just bigger ones! • Allow creative flexibility: teams can decorate manually after coding. • Keep a live scoreboard if you're running a competitive mode!
Learning Outcomes and Skills:	Structural design, sustainable planning Block-based coding, automation Creative expression, design thinking Resource and space optimization Renewable energy concepts
Feedback Collection (10 minutes):	• Quick exit ticket: 3-2-1 reflection (3 things learned, 2 challenges, 1 thing they want to improve) • Optional: Google Form survey with 3 quick questions: "What was your favorite part of the Eco-City challenge?" "What part of coding the agent was hardest?" "How would you improve your city next time?"
Assessment & Evaluation:	Completion of Resource Gathering 10 Points House Construction with Code 10 Points Installation of Renewable Energy 10 Points Artistic Park Creation 10 Points Efficient use of Coding Structures 10 Points

	Overall Eco-City Design	10 Points
Conclusion (5 minutes):	• Recap what STEAM skills were used. • Highlight great examples from students' cities. • Encourage students to think about how coding and art together can solve real-world problems. • Invite them to improve their cities later or create new ones!	
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1FQgcDTkFJohAw9jHK7RRju_XIS1FRiW21nLMhZO9Wns/edit?tab=t.0	
Game Link:	https://makecode.com/_4zW8K3VL7EyT	

RĪGAS
64.VIDUSSKOLA

Rigas 64 VIDUSKOLA Learning Scenarios

Learning Scenario 1: Algodoo Contraption Challenge: The Amazing Chain Reaction Machine	
Target Groups:	• Age: 10–15 years old (Middle school to early high school) • Level: No prior Algodoo experience needed. Basic computer literacy assumed. • Group Size: Small teams (2–3 students)
Learning Objectives:	• **Physics Principles:** Understand and apply concepts like gravity, momentum, friction, levers, pulleys, collisions, and energy transfer (potential to kinetic). • **Engineering Design:** Practice the engineering design process (ideate, build, test, iterate, improve) to create a working chain reaction. • **Computational Thinking:** Develop logical sequencing and problem-solving skills by planning how one action will trigger the next. • **Creativity & Innovation:** Design unique and imaginative chain reaction steps using Algodoo's tools. • **STEAM Integration:** Combine physics (Science), design/construction (Engineering/Technology), visual aesthetics (Art), and spatial reasoning/measurement (Math).
Materials & Resources:	• Computers with Algodoo installed (free download from algadoo.com). • Projector or smartboard for demonstration. • Printed "Algodoo Chain Reaction Challenge Sheet" (see Phase 2 below). • Optional: Paper for sketching initial ideas.
Introduction (15 minutes):	• **Hook:** Show a short, engaging video of a Rube Goldberg machine or a complex Algodoo chain reaction. Ask: "How cool was that? What made it work?" • **Context:** Introduce Algodoo as a "2D physics playground" where they can build almost anything and see how physics makes it move. Explain they'll be "Contraption Engineers" today. • **Relevance:** Connect chain reactions to real-world examples (dominoes, assembly lines, cause-and-effect in nature or systems) and the fun of problem-solving. • **Algodoo Quick Tour:** Briefly demonstrate the Algodoo interface:

	• Drawing tools (box, circle, polygon, plane, gear, spring, rope/chain). • Interaction tools (drag, scale, rotate). • Material tool (changing friction, density, restitution/bounciness). • Joints (hinge/axle, fixate). • Play, Pause, Reset buttons. • The "Eraser" and "Knife" tools. • **Team Formation:** Divide students into teams. Suggest roles: • **Lead Designer:** Sketches initial ideas and oversees the overall look. • **Mechanism Specialist:** Focuses on how individual parts will connect and function. • **Chief Tester:** Runs simulations frequently, identifies problems, and suggests fixes. (Encourage collaboration and role-sharing).

Scenario Presentation (Step-by-Step)

1. Mission Briefing (5 minutes):

- "Welcome, Contraption Engineers! The 'World of Wacky Inventions' expo is looking for the most ingenious chain reaction machine. Your team's mission is to design and build a machine in Algodoo that performs at least 3 distinct sequential actions, starting with a single click or drop, to achieve a final simple goal (e.g., pop a balloon, knock over a specific block, raise a flag)."

2. Hand out the Challenge Sheet (2 minutes):

- Distribute the "Algodoo Chain Reaction Challenge Sheet."
- Briefly review the challenges and the "Machine Blueprint" section.

3. Walkthrough Key Tools (8 minutes):

- Based on the first few challenge ideas, quickly demo how to create a simple lever, make something fall and hit another object, or use a hinge.

4. Launch Algodoo! (2 minutes):

- Students open Algodoo and create a new scene.

5. Challenge Time (45-60 minutes):

- Students work through the challenges on their sheet, designing and building their chain reaction.
- Teacher circulates, offering guidance, asking probing questions ("What happens if that ramp is steeper? What if the ball is heavier?"), and troubleshooting basic Algodoo usage.

6. Showcase & Celebration (10-15 minutes):

• Each team presents their chain reaction machine to the class. They should explain the steps and what physics principles they think are at play. • Run each machine! Celebrate successes and clever solutions, even if some parts don't work perfectly every time (that's engineering!).	
Teacher Tips and Strategies:	• **Start Simple:** Encourage them to get one action working before moving to the next. • **Iterate Often:** Test, test, test! Small adjustments can make a big difference. Ctrl+Z (undo) is their friend. • **Material Matters:** Prompt them to experiment with different materials (wood vs. steel, high vs. low friction). • **Embrace "Failure":** When something doesn't work, it's a learning opportunity. "Why didn't that ball roll far enough?" • **Keep the Goal in Mind:** What is the *final* action they want their machine to perform?
Learning Outcomes and Skills:	• **Physics Application:** Demonstrated understanding of gravity, momentum, etc. • **Engineering Process:** Evidence of planning, building, testing, and revising. • **Problem-Solving:** Overcoming challenges in making their machine work. • **Creativity:** Novelty in their chain reaction steps. • **Algodoo Proficiency:** Competent use of basic tools.
Feedback Collection (10 minutes):	• **Group Discussion: "**What was the trickiest part of your machine to get working?" "What physics idea did you see in action that surprised you?" "If you had more time, what would you add or change?" • **Optional:** Students can take a screenshot of their favorite part of their machine and write one sentence explaining why it's cool or how it works.
Assessment & Evaluation:	• **Formative Observation:** Engagement, collaboration, use of physics vocabulary, problem-solving attempts. • **Challenge Sheet Review:** Check their "Machine Blueprint" for planning and completion of steps. • **Machine Functionality:** Does the chain reaction achieve at least 2-3 distinct, sequential steps initiated by a single action? Does it attempt to reach the final goal?

	Presentation: Ability to explain their machine's steps and design choices.
Conclusion (5 minutes):	• • Congratulate them on their ingenious contraptions! • Reiterate how physics principles govern the world around us, from simple toys to complex machines. • Emphasize that the process of designing, testing, and improving is key to all engineering and scientific discovery. • Encourage them to continue exploring Algodoo to build other fun simulations or solve new challenges.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/189zJmMkwFcqv-FEB-k1p5aZi-rti7n9vNu3u7JN50Gs/edit?usp=sharing
Game Link:	https://www.algodoo.com/

Learning Scenario 2: Probability Playground – Designing a Carnival Game in Polypad	
Target Groups:	○ **Age:** 12–18 years old (Middle school to high school). ○ **Level:** No prior Polypad or advanced math experience needed. Basic computer literacy assumed. ○ **Group Size:** Individual work (1 student per device, with peer playtesting).
Learning Objectives:	○ **Mathematics (Probability):** Understand the difference between theoretical probability (what *should* happen) and experimental probability (what *actually* happens). ○ **Game Design:** Design a game with clear rules, win conditions, and variable difficulty (fair vs. "rigged" games). ○ **Computational / Data Skills:** Use digital manipulatives to generate, record, and visualize large sets of data instantly. ○ **STEAM Integration:** Combine Mathematics (probability/data) with Technology (digital simulation) and Art (visual design of the game board).
Materials & Resources:	○ Computers or tablets with internet access (Polypad runs free in the browser at polypad.amplify.com). ○ Projector or smartboard for demonstration. ○ Printed "Polypad Carnival Game Challenge Sheet" (see below).

Introduction (15 minutes):	**Hook: Ask the class:** "Have you ever played a carnival game and felt like it was rigged so you couldn't win? Today, *you* are the carnival game designers, and you get to decide how fair your game is!" **Context:** Introduce Polypad as a digital canvas where math comes to life. **Polypad Quick Tour:** Briefly demonstrate the Polypad interface: • **The left-hand menu (specifically the Probability and Data section).** • **How to drag dice, coins, and spinners onto the canvas.** • **The magical "Roll" and "Spin" buttons, and the slider to roll 10 or 100 times at once!** **Solo Setup:** Explain that each student will work independently to invent a game, set the rules, and simulate playing it.
Scenario Presentation (Step-by-Step)	
Mission Briefing (5 minutes): **Mission Briefing (5 minutes):** "Welcome to the Mathigon Carnival! Your mission is to invent a new, interactive game of chance using dice and spinners. You will set the rules, run the simulation, and use data charts to prove if your game is an easy win or a tough challenge." **Hand out the Challenge Sheet (2 minutes):** Distribute the "Polypad Carnival Game Challenge Sheet." **Walkthrough Key Tools (8 minutes):** Demonstrate how to pull out a custom spinner and change its colors/wedges. Show them how to connect a spinner to a Bar Chart to track the spins. **Launch Polypad! (2 minutes):** Students go to polypad.amplify.com and click "Open Polypad". **Challenge Time (40-50 minutes):** Students work through the challenges, designing and testing their games. The teacher circulates to help with tool navigation and prompt mathematical thinking ("Do you think Red will win more often than Blue? Let's test it!"). **Showcase & Celebration (10-15 minutes):** Students swap seats. They must read the rules of their neighbor's game and "play" it by clicking the roll/spin buttons to see if they can win!	
Teacher Tips and Strategies:	**The Power of Simulation:** Make sure students know they don't have to click the spinner 100 times manually. Show them the "Spin 100x" button on the bottom menu when the spinner is selected. **Embrace the "Rigged" Game:** It is perfectly fine (and actually highly educational) if a student designs a game that is almost impossible to win, as long as they realize it when looking at the data!
Learning Outcomes and Skills:	**Mathematics & Probability:** Differentiating between theoretical probability (what *should* happen) and experimental probability (what *actually* happens over 100 spins). **Data Analysis:** Successfully collecting, visualizing (using charts), and interpreting large sets of data. **Game Design Thinking:** Creating clear rules and balancing game mechanics to make a game either "fair" or intentionally difficult. **Problem-Solving:** Adjusting game difficulty based on the results of their data simulations.

	Digital Literacy: Navigating digital manipulatives and using computer simulation to test theories quickly.
Feedback Collection (10 minutes):	**Group Discussion: Ask the class:** • "Whose game was the hardest to win, and how did your data prove it?" • "Did your chart (experimental data) exactly match what you thought would happen just by looking at the spinner?" • "If you wanted the player to win exactly 50% of the time, how would you change your rules or your spinner?" **Exit Ticket (optional):** Have students write down: "One thing I learned about probability today:" and "One way I would change my carnival game to make it more fun:"
Assessment & Evaluation:	**Formative Observation:** Observe student engagement with the digital tools, their testing process, and how they interact during the peer showcase. **Challenge Sheet Review:** Check their completed "Polypad Carnival Game Challenge Sheet" (Darbalapa) to ensure they successfully recorded their data and thoughtfully answered the reflection questions. **Game Design & Functionality:** Does the game have clear, logical rules? Did the student successfully link the spinner to the data chart to visualize their game's difficulty? **Brief Showcase:** Assess the student's ability to explain *why* their game is fair or "rigged" using the vocabulary of math and data.
Conclusion (5 minutes):	Congratulate them on becoming data-driven Game Designers! Reiterate that real game developers (from carnival game creators to massive video game studios) use math and probability every single day to balance their games and make them fun (or profitable). Encourage them to keep exploring Polypad's tools, or challenge them to try coding their new game rules into Scratch later. Highlight the STEAM connection: Today they combined mathematical data, game design, and digital technology into one project.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1dy_GULQsgY1_Wy3k3SqNFGJDddq0J_AG/edit?usp=sharing&ouid=115364178447004618296&rtpof=true&sd=true
Game Link:	https://polypad.amplify.com

Learning Scenario 3: Code Quest: Designing an Interactive Maze Game in Scratch	
Target Groups:	• Age: 10–15 years old (Middle school or early high school) • Level: Basic computer literacy; no prior programming experience required. • Group Size: Small teams (2–3 students)
Learning Objectives:	• Understand and apply basic programming concepts (events, motion, loops, conditionals, variables) using Scratch. • Develop computational thinking skills: decomposition, pattern recognition, algorithmic thinking, debugging. • Design an interactive game, including player controls, obstacles, and goals. • Foster creativity through custom sprite/background design and game mechanics. • Collaborate effectively in a team to plan, build, and test a digital project. • Integrate Art (sprite/maze design) and Math (coordinates) with Technology & Engineering (coding).
Materials & Resources:	• Computers or tablets with internet access and Scratch (online at scratch.mit.edu or offline editor). • Projector or smartboard for demonstration. • Printed "Scratch Maze Crafters Challenge Sheet" (see Phase 2 below). • Optional: Paper for sketching maze designs.
Introduction (15 minutes):	• **Hook:** Show a very simple, fun Scratch maze game (or a GIF of one). Ask: "Ever wanted to build your own game? Today, you're game designers!" • **Context:** Introduce Scratch as a visual programming language that makes coding like building with digital LEGOs. Explain they'll be undertaking the "Maze Crafters Challenge." • **Relevance:** Briefly mention how game development uses coding, art, storytelling, and logic – all key STEAM skills. • **Scratch Quick Tour:** Briefly demonstrate the Scratch interface: Stage, Sprites pane, Blocks Palette (Motion, Looks, Events, Control, Sensing, Variables). Show how to drag and snap blocks. • **Team Formation:** Divide students into teams. Suggest roles: • **Lead Coder:** Focuses on implementing the core logic.

	• **Art Director:** Designs/chooses the maze and sprites. • **Quality Assurance (QA) Tester:** Plays the game frequently to find bugs and suggest improvements. (Encourage role sharing/rotation if time/interest allows).

Scenario Presentation (Step-by-Step)

1. Mission Briefing (5 minutes):

- "Welcome, Maze Crafters! A new gaming company, 'Pixel Perfect Puzzles,' needs fresh talent. Your team's mission is to design and code an exciting maze game. You'll build everything from the ground up: the maze, the hero, the dangers, and the rewards!"

2. Hand out the Challenge Sheet (2 minutes):

- Distribute the "Scratch Maze Crafters Challenge Sheet."
- Briefly point out the challenges and the "Game Features Checklist."

3. Walkthrough Key Blocks (8 minutes):

- For the first 1-2 challenges, demonstrate the *types* of blocks they'll need (e.g., "for movement, you'll look in 'Events' and 'Motion'"). Don't solve it for them, just orient them.

4. Launch Scratch! (2 minutes):

- Students open Scratch and create a new project.

5. Challenge Time (40-50 minutes):

- Students work through the challenges on their sheet, using Scratch.
- Teacher circulates, facilitating, asking guiding questions, and troubleshooting.

6. Playtest & Showcase (10-15 minutes):

- Teams get a chance to play each other's games.
- Each team briefly shows their game to the class, highlighting one feature they are proud of or one challenge they overcame.

Teacher Tips and Strategies:

- **Model Sparingly:** Demonstrate concepts, not entire solutions. E.g., show how one arrow key moves a sprite, then let them figure out the others.
- **Chunk It:** The Challenge Sheet naturally chunks tasks. Remind them to focus on one challenge at a time.
- **"Bug Hunting" is Normal:** Frame errors as "bugs" to find and fix – a normal part of coding. Encourage systematic testing.
- **Promote "Rubber Ducking":** If a team is stuck, have them explain their code (and what they want it to **do) to each other (or a "rubber duck") before asking the teacher.**

Encourage Customization: After core mechanics, let them personalize sounds, colors, sprite appearances.

Learning Outcomes and Skills:	• Coding Fundamentals: Using events, motion, conditionals, variables. • Problem-Solving: Debugging, logical sequencing. • Design Thinking: Planning game flow, user experience. • Creativity: Customizing art and game elements. • Collaboration: Team communication and shared building.
Feedback Collection (10 minutes):	• **Group Discussion:** "What was the most challenging coding block to use and why?" "What part of your game are you most proud of?" "If you had more time, what would you add?" • **Exit Ticket (optional):** "One new Scratch trick I learned:" and "One thing I'd like to try coding next:"
Assessment & Evaluation:	• **Formative Observation:** Engagement, collaboration, problem-solving approaches. • **Challenge Sheet Review:** Check their "Game Features Checklist" for completion. • **Game Functionality:** Does the game work as intended for the completed challenges? ◦ *(Simple Rubric: Sprite moves, Walls block, Goal is reachable, Collectibles/Hazards work if implemented).* • **Brief Showcase:** Assess ability to explain a feature of their game.
Conclusion (5 minutes):	• Congratulate them on becoming game developers! • Reiterate that the logic (ifs, loops, sequences) they used is the foundation of all programming. • Encourage them to keep exploring Scratch, try more complex projects, or share their games online. • Highlight the power of combining code, art, and imagination (STEAM).
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1PX4PMx4fo71I4xlgEq0Z7Wo1OoVz9lKS/edit?usp=sharing&ouid=115364178447004618296&rtpof=true&sd=true
Game Link:	https://scratch.mit.edu/

Learning Scenario 4: Dive into Density – Exploring Mass and Volume	
Target Groups:	Age: 12–16 years old (Middle or early high school) Level: Basic math and science skills (mass, volume, division) Group Size: Individual or small groups (2–3 students)
Learning Objectives:	• Define density as mass divided by volume. • Explore the relationship between mass, volume, and density using interactive simulation. • Compare densities of different objects and substances (e.g., wood, aluminum, brick). • Predict whether an object will float or sink based on its density. • Use measurements to calculate and compare density values. • Enhance inquiry skills through guided exploration and experimentation.
Materials & Resources:	• Computers or tablets with internet access • Projector or smartboard for demonstration • **Access to:** https://phet.colorado.edu/en/simulations/density • Printed or digital 'Density Simulation Worksheet' • **Calculator (optional)**
Introduction (15 minutes):	• **Hook:** Show a picture or video of objects floating and sinking in water (e.g., wood vs. metal). **Ask:** 'Why do some things float and others sink?' • **Context:** Introduce the concept of density as a way to understand how tightly matter is packed. • **Relevance:** Explain real-world applications – shipbuilding, materials science, and scuba diving. • **Quick overview of the PhET Density simulation:** How to select objects, measure mass and volume, and observe floating behavior.
Scenario Presentation (Step-by-Step)	
• **1. Mission Briefing (5 minutes):** 'Welcome to the Density Lab! Today you're scientists exploring what makes things float or sink. You'll test materials and calculate their densities.' • **2. Distribute Worksheet (2 minutes):** Hand out the worksheet that includes prediction tasks, tables for recording data, and analysis questions.	

• **3. Simulation Walkthrough (5 minutes):** Demonstrate how to measure mass and volume using the PhET tools, and how to calculate density (mass ÷ volume). • **4. Inquiry Time (30–35 minutes):** Students explore the simulation, collect data for several objects, and answer guided questions. • **5. Group Reflection (10 minutes):** Discuss findings. Which objects were surprising? What patterns did they observe about floating and density?	
Teacher Tips and Strategies:	• Emphasize making and testing predictions before checking the results. • **Encourage students to use the equation:** Density = Mass ÷ Volume. • Foster exploration by letting students choose their objects to test. • **Ask guiding questions like:** 'What do you notice about the objects that float?' • **Use real-life analogies:** Oil vs. water, metal vs. plastic.
Learning Outcomes and Skills:	• **Scientific Concepts:** Mass, volume, density, buoyancy. • **Mathematical Application:** Division, units, ratios. • **Data Collection and Analysis:** Observation and measurement. • **Critical Thinking:** Comparing, predicting, and interpreting results. • **Scientific Inquiry:** Hypothesis testing and evidence-based conclusions.
Feedback Collection (10 minutes):	**Group Discussion:** • Which object surprised you most? • Did any of your predictions turn out wrong? Why? • One new thing I learned about density is: _____
Assessment & Evaluation:	• **Formative Observation:** Participation, use of simulation tools, and reasoning. • **Worksheet Review:** Accuracy of recorded measurements and calculated densities. • **Reflection Questions:** Depth of understanding and application of concepts.

Conclusion (5 minutes):	• **Summarize key ideas:** Density explains why things float or sink. • **Encourage curiosity:** Try testing objects at home or in real water. • Highlight how virtual labs can help us learn science safely and visually.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/18xUAomg6-vnizWkkkwV28ZXSBYUTB1yx/edit?usp=sharing&ouid=101122000948706492484&rtpof=true&sd=true
Game Link:	https://phet.colorado.edu/en/simulations/density

Learning Scenario 5: Cell Explorers: The Organelle Discovery Mission	
Target Groups:	• Age: 12–16 years old (Middle or early high school) • Level: Basic biology knowledge; familiarity with cells is helpful Group Size: Individuals or pairs
Learning Objectives:	• Identify and understand the function of major organelles in plant and animal cells. • Distinguish between structures in plant and animal cells. • Use an interactive simulation to explore cell biology concepts. • Apply scientific vocabulary related to cells and organelles. • Engage in inquiry-based learning and develop observation skills. Reflect on learning through written responses and group discussion.
Materials & Resources:	• Computers or tablets with internet access • Projector or smartboard for demonstration • Access to: https://www.biomanbio.com/HTML5GamesandLabs/Cellgames/cellexplorerpagehtml5.html • Printed 'Cell Explorer Challenge Sheet' Notebook or paper for notes and sketches (optional)

Introduction (15 minutes):	• Hook: Display an image of a cell (plant/animal) with labels covered. Ask: 'What do you recognize?' • Context: Explain that cells are the basic units of life and contain organelles with specific functions. • Relevance: Mention the importance of understanding cells for biology, health, and life sciences. • Tool Overview: Show how the Cell Explorer game works – drag and drop organelles, read info, take a quiz. • Group Formation: Have students work alone or in pairs, depending on available devices.

Scenario Presentation (Step-by-Step)

Mission Briefing (5 minutes):

'Welcome, Cell Explorers! You are entering a microscopic world. Your task is to identify organelles and learn how they work to keep the cell alive!'

Hand out the Challenge Sheet (2 minutes):

Distribute the worksheet. Explain: they will record the name, function, and location of each organelle.

Walkthrough First Organelle (5 minutes):

Briefly demonstrate how to drag one organelle, view its info, and note its function.

Begin Cell Quest! (25–30 minutes):

Students interact with the game, complete the drag-and-drop activity and quiz.

Reflection & Quiz Discussion (10–15 minutes):

Students share their quiz results and discuss any organelles that were difficult to remember

Teacher Tips and Strategies:	Demonstrate how to interact with the game, but don't provide all answers. Guide students by asking: 'What happens if this organelle stops working?' Encourage students to connect structures to functions (e.g., mitochondria = energy). Highlight differences between plant and animal cells (e.g., cell wall, chloroplasts). Reinforce vocabulary: challenge students to use scientific terms when explaining.

Learning Outcomes and Skills:	• Biology Knowledge: Cell parts and their functions. • Scientific Thinking: Observation and classification. • Digital Literacy: Using online tools for learning. • Language Development: Scientific vocabulary use. • Reflection: Analyzing and discussing understanding.
Feedback Collection (10 minutes):	Group Discussion: • Which organelle was the easiest or hardest to remember? • What did you learn that surprised you? • One question I still have about cells: _____
Assessment & Evaluation:	• Formative Observation: Monitor participation, interaction with the tool, and vocabulary use. • Challenge Sheet Review: Check completed answers for accuracy. • Quiz Score Review: Use quiz performance as a formative indicator of learning. • Reflection Sharing: Assess ability to articulate what they learned and found challenging.
Conclusion (5 minutes):	• Celebrate completion of the Cell Explorer challenge. • Reinforce the concept that every cell is a complex, organized system. • Encourage students to keep exploring cells and biology using interactive resources.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/18xUAomg6-vnizWkkkwV28ZXSBYUTB1yx/edit?usp=sharing&ouid=101122000948706492484&rtpof=true&sd=true
Game Link:	https://www.biomanbio.com/HTML5GamesandLabs/Cellgames/cellexplorerpagehtml5.html

Lesson Scenario 6: Code Quest: Building an Interactive Europe Map Quiz	
Target Groups:	• Age: 10–14 years old (Middle school) • Level: Basic computer literacy; no prior programming experience required. • Group Size: Individual work (1 student per device)
Learning Objectives:	• Geography: Identify European countries and recall their capital cities. • Coding: Understand and apply events (clicking sprites), simple conditionals (if/else for answer checking), and variables (scoring). • Design: Create an interactive interface where clicking a location triggers a question. • Computational Thinking: Decompose a quiz into a repeating pattern (Question -> Input -> Check Answer -> Feedback).
Materials & Resources:	• Computers with internet access to scratch.mit.edu. • Projector for demonstration. • **Map Asset:** An image of a blank map of Europe (or students can use the "Europe" sprite/backdrop available in Scratch if applicable, or upload one). • List of European countries and capitals (handout or on whiteboard).
Introduction (15 minutes):	• **Hook:** "How well do you know Europe? Today, instead of taking a quiz, you are going to *build* one to stump your friends!" • **Context:** Explain that interactive maps are used everywhere (Google Maps, travel apps). We will build a simple version where clicking a country asks you to name its capital. • **Scratch Quick Tour:** Briefly remind them of the interface. Point out the **Sensing** blocks (specifically ask [ ] and wait and answer) and **Control** blocks (if/then/else), as these are the tools for a quiz. • **The Mission:** You will create a game where players click on a country, type the capital city, and get points for correct answers.
Scenario Presentation (Step-by-Step)	
• **1. Mission Briefing (5 minutes):** • Your goal is to create a quiz with at least 5 different European countries. • You need a "Game Master" (a sprite that asks the questions) and clickable "Hotspots" (sprites placed over countries). • **2. Setting the Stage (5 minutes):** • Backdrop: Have students upload a map of Europe as the Backdrop. • The Game Master: Choose a character (e.g., "Abby" or "Gobo") to be the host who gives feedback ("Correct!" or "Try Again"). **3. Walkthrough: The "Hotspot" Technique (10 minutes):** • Teacher Demo: Explain that we can't easily click just one part of a background image. The	

trick is to create invisible (or visible) sprites to act as buttons.

- Step A: Paint a new Sprite. Draw a small dot or circle.
- Step B: Place this dot over a country (e.g., France).
- Step C: Name the sprite "France Button".
- Ask students: "What event block do we need if we want something to happen when we click this dot?" (Answer: When this sprite clicked).

4. Challenge Time (40-50 minutes):

- Phase 1: The Question Script: Students code the first country button.
- When this sprite clicked
- Ask [What is the capital of France?] and wait (Sensing)
- If <(answer) = [Paris]> then
- Say [Correct!] for 2 seconds
- Else
- Say [Wrong! It is Paris.] for 2 seconds
- Phase 2: Scale Up: Duplicate the button sprite, move it to a new country (e.g., Germany), and change the code (Question: Germany? Answer: Berlin). Repeat for 5 countries.

Phase 3: Scoring (Extension): Create a Variable called "Score". Add Change [Score] by (1) inside the "If Correct" part of the script.

5. Playtest & Showcase (10 minutes):

- **Students swap computers.**
- **Challenge: Can the tester get a perfect score on their neighbor's map?**

Teacher Tips and Strategies:	• **Invisible Buttons:** For a cleaner look, students can set the "Ghost Effect" of the button sprites to 100. The sprite is still clickable, but invisible! • **Spelling Matters:** Remind students that Scratch is strict about spelling. If the player types "paris " (with a space), it might count as wrong. (Advanced tip: Show them how to just check if the answer *contains* "Paris"). • **Chunking:** Don't let them try to do all 5 countries at once. Make one country work perfectly first, then duplicate. • **Geography Helper:** Keep a list of capitals on the board so coding isn't slowed down by geography knowledge gaps during the build phase.

Learning Outcomes and Skills:	**1. Geography Knowledge & Skills**: • Identify and locate: Accurately identify and click on specific European countries on a map based on a textual prompt. • Geographical awareness: Develop a better understanding of the relative positions of major European countries. **2. Scratch Programming Concepts**: • Events: Utilize when Green Flag clicked, when this sprite clicked, and when I receive [broadcast] blocks to control program flow and sprite interactions. • Variables: Create and manipulate a global variable (TargetCountry) to manage the game's state (which country is currently being sought) and a Score variable to track player progress. • Conditionals: Implement if/else statements to evaluate player actions (sprite clicks) against the TargetCountry variable, providing different responses for correct and incorrect answers. • Broadcasting: Employ broadcast messages (Next Level) to enable efficient communication and synchronization between different sprites (e.g., the Host asking a new question after a correct answer). • Sensing & Looks: Use say blocks for displaying questions and feedback, start sound blocks for auditory feedback, and potentially hide/show or ghost effects for sprite visibility. • Sprite Management: Create, position, and name multiple interactive sprites (country "buttons") effectively to represent clickable locations on a map. **3. Computational Thinking Skills:** • Decomposition: Break down the complex task of creating an interactive quiz into smaller, manageable sub-problems (e.g., asking a question, registering a click, checking an answer, updating the score, moving to the next question). • Pattern Recognition: Identify repeatable patterns in the code (e.g., the similar when this sprite clicked scripts for each country button, the question-answer-feedback loop). • Abstraction: Understand how the TargetCountry variable serves as an abstraction for the current country to be found, making the game logic more flexible. • Debugging: Identify and troubleshoot errors in their code, particularly those related to variable consistency (matching TargetCountry to sprite names), broadcast messages, and conditional logic. **4. Game Design & User Interface (UI):** • Interactive Design: Design and implement an interactive element (clickable sprites on a map) for an educational game. • Feedback Mechanisms: Plan and implement clear visual and auditory

	feedback (e.g., correct/incorrect sounds, score updates) to guide the player.
Feedback Collection (10 minutes):	**Exit Ticket:** Method: A quick, short response collected at the end of the lesson. Example Prompts: "One thing I learned about European geography today is..." "One Scratch block I found useful today was _____ because..." "I am still a bit confused about _____."
Assessment & Evaluation:	**Game Functionality:** • Does clicking a country trigger a question? (Yes/No) • Does the game correctly identify the right answer? (Yes/No) • Does the game give feedback for a wrong answer? (Yes/No) **Code Review:** Check if they used the Ask and Wait block and the If/Else block correctly. **Plenary Question**: "Why did we use If/Else instead of just If?" (Answer: Because we wanted to do something specific when the answer was wrong, too).
Conclusion (5 minutes):	Congratulate them on combining Geography and Computer Science. Homework/Extension Idea: "How could you make this harder? Maybe the map disappears and you have to guess where the country is?"
Link for Challanges & CodeBlocks	https://drive.google.com/drive/folders/1T8lLxLJg068QtoCci8BeH2YUubfrQ3DY
Game Link:	https://scratch.mit.edu/

SREDNJA ŠKOLA METKOVIĆ

Srednja skola Metkovic Learning Scenarios

Learning Scenario 1: Cities Minecraft scenario – Education Edition	
Target Groups:	Students aged 15-18
Learning Objectives:	• Understand core concepts of **urban sustainability** and **resource management** • Recognize interconnections between **infrastructure, environment, population health, and energy** • Apply **critical thinking** in city development under resource constraints • Explore **green technologies**, pollution control, traffic planning, and zoning • Develop **soft skills**: collaboration, negotiation, scenario-based decision-making • Introduce concepts of **circular economy**, **eco-innovation**, and **public policy design** • Practice using **data visualization tools** for monitoring pollution, noise, traffic, and citizen satisfaction
Materials & Resources:	• Computers with **Minecraft Education Edition** • Projector/Smartboard to present successful city designs or discuss failure cases • **Urban Planner Logbook** handout (for recording design ideas, decisions, and impact) • Printed **Challenge Briefing Sheets** • Optional: Internet access for research on sustainable cities (e.g., Copenhagen, Singapore)
Introduction (15 minutes):	• **Hook:** Show real-life short video clips about urban smog, traffic congestion, green buildings, and city farming • **Context:** "You are urban planning teams hired to design a new eco-city for 100,000 citizens. Your job? Make it livable, clean, efficient, and green!" • **Relevance:** Connect simulation goals to UN SDGs (Sustainable Development Goals), especially #11: Sustainable Cities and Communities • **Tool Familiarization:** Quick demonstration of designing zones, creating infrastructure, energy systems, water management, environmental measures, observation tools • **Team Formation:** Assign student roles: • **Sustainability Strategist** (focus on environment and energy) • **Infrastructure Planner** (roads, transport, zoning) • **Citizen Advocate** (services, health, education, happiness)
Scenario Presentation (Step-by-Step)	
• **Mission Briefing (5 minutes):** • "Design a self-sufficient, environmentally friendly city with minimal pollution and traffic congestion within 60 in-game days." • **Distribute Logbooks (2 minutes):** • Students use logbooks to plan zones, utilities, renewable energy strategies, etc. • **Simulation Start (5 minutes):** • Teams begin laying out their cities. Set goals: low pollution, renewable energy %, citizen happiness indicators. • **Challenge Time (30–40 minutes):**	

• Students play the game in teams, making real-time decisions and adjusting to simulated problems (e.g., resource shortages, insufficient food supplies, environmental degradation). • **Crisis Scenarios (Optional, 10 minutes):** • Introduce surprise challenges: natural disasters, budget cuts, energy shortages. Teams must respond live. • **City Showcase & Debrief (10–15 minutes):** • Teams present city screenshots, explain major strategies, and share what worked or failed	
Teacher Tips and Strategies:	• **Promote reflection:** "What would be the real-world consequence of your city design?" • **Guide inquiry:** "What causes traffic congestion? How does zoning affect the environment?" • **Differentiate challenges:** • *Beginner task:* Survive 60 days with no blackouts • *Advanced challenge:* Reach 80% renewable energy use while maintaining citizen happiness indicators > 75%
Learning Outcomes and Skills:	• **Environmental Literacy:** Understand emissions, waste cycles, green energy • **Systems Thinking:** See how different city parts interact (e.g., traffic ↔ pollution ↔ health) • **Digital Literacy:** Operate a complex simulation environment • **Civic & Global Competence:** Grasp real-world urban challenges through simulation • **Communication & Leadership:** Work in diverse roles within a team setting
Feedback Collection (10 minutes):	• **Group Reflection:** What trade-offs did your team make? • **Exit Ticket:** "If this were your real city, what would you prioritize next?"
Assessment & Evaluation:	• **Formative:** Observe team collaboration, decision-making, problem resolution • **Summative:** • Review team logbooks for strategy notes and adaptations • Screenshot analysis: Are traffic, energy, and citizen needs balanced? • Optional Rubric: Green energy % | Pollution levels | Happiness score | Resilience to crises
Conclusion (5 minutes):	• Celebrate sustainable city successes and "beautiful failures" • Discuss how gaming can simulate complex, real-world systems • Encourage students to explore careers in green tech, urban design, public policy • Offer links to real-life smart city projects and open urban data portals
Link for Challenges & CodeBlocks	https://docs.google.com/document/d/1vUql53GMrpFpysQNLTwM-yktOOE628lBKIUHT8-ein8/edit?usp=sharing
Game Link:	https://education.minecraft.net/en-us

Learning Scenario 2:"Volcanoes in Minecraft: Erupting Knowledge and Understanding"	
Target Groups:	9th and 10th Grade Geography Students
Learning Objectives:	• Students will identify and describe the different types of volcanoes and their formation processes. • Students will analyze the impact of volcanic eruptions on the surrounding environment and human settlements. • Students will model and simulate volcanic activity using Minecraft Education Edition's tools and features. • Students will understand the geological processes involved in plate tectonics and their relationship to volcanism. • Students will develop spatial reasoning and problem-solving skills through Minecraft-based activities. • Students will effectively communicate their findings and interpretations through presentations and collaborative projects.
Materials & Resources:	• Computers/tablets with Minecraft Education Edition installed. • Minecraft Education Edition world files pre-designed with volcanic landscapes and scenarios. • Information resources on volcanoes (textbooks, online articles, videos, geological maps). • Handouts with guided questions, research prompts, and assessment rubrics. • Whiteboards or digital collaborative tools for brainstorming and planning. • Minecraft Education Edition camera and portfolio features for documentation
Introduction (15 minutes):	• Begin with a captivating video or image showcasing a volcanic eruption and its effects. • Discuss the global distribution of volcanoes and their relationship to plate tectonics. • Introduce the concept of different volcano types (shield, stratovolcano, cinder cone) and their characteristics. • Explain how Minecraft Education Edition can be used to model and explore volcanic phenomena. • Briefly demonstrate the Minecraft Education Edition world created for the lesson.
Scenario Presentation (Step-by-Step)	

Day 1: Volcano Formation and Types (45 minutes)

- Students explore the Minecraft world, identifying different terrain features and volcanic structures.
- Students research and document the formation processes of various volcano types using Minecraft's camera and portfolio options.
- Students build models of different volcano types using Minecraft blocks and materials.
- Assign homework: Research plate tectonics and its relationship to volcanic activity.
- **Day 2: Simulating Volcanic Eruptions (45 minutes)**
- Students use Minecraft's command blocks and other tools to simulate volcanic eruptions with varying degrees of intensity.
- Students analyze the impact of simulated eruptions on the surrounding Minecraft environment (e.g., lava flow, ash clouds).
- Students document and present their simulation results, discussing the factors that influence eruption severity.
- **Day 3: Environmental and Human Impact (45 minutes)**
- Students design and build Minecraft scenarios depicting the effects of volcanic eruptions on human settlements and ecosystems.
- Students research real-world examples of volcanic disasters and incorporate their findings into their Minecraft scenarios.
- Students create presentations or videos showcasing their Minecraft scenarios and discussing the challenges of living near volcanoes.
- **Day 4: Plate Tectonics and Volcanic Distribution (45 minutes)**
- Students create a Minecraft model of the Earth's tectonic plates, highlighting plate boundaries and volcanic hotspots.
- Students use Minecraft to show the different types of plate boundries, and how each type creates volcanoes.
- Students present their models, explaining the relationship between plate tectonics and the global distribution of volcanoes.

Teacher Tips and Strategies:	• Provide clear instructions and guidelines for Minecraft activities. • Encourage collaboration and teamwork among students. • Integrate real-world data and examples into the Minecraft simulations. • Use Minecraft's camera and portfolio features for documentation and assessment. • Facilitate class discussions and encourage critical thinking.

	• Use command blocks sparingly, and provide pre-made command sequences if needed. • Encourage creative problem solving.
Learning Outcomes and Skills:	• Improved understanding of volcanic processes and their impact. • Enhanced spatial reasoning and problem-solving skills. • Development of digital literacy and Minecraft proficiency. • Improved communication and presentation skills. • Increased collaborative and teamwork skills. • Better understanding of plate tectonics.
Feedback Collection (10 minutes):	• Conduct a class discussion to gather student feedback on the Minecraft activities. • Use a short survey or questionnaire to assess student engagement and learning. • Review student portfolios and presentations for insights into their understanding.
Assessment & Evaluation:	• Assessment of Minecraft models and simulations. • Evaluation of student presentations and research projects. • Grading of written reflections and documentation. • Assessment of participation in class discussions and collaborative activities. • Rubrics that include the accuracy of the models, the quality of the research, and the clarity of the presentations.
Conclusion (5 minutes):	• Summarize the key learning outcomes and geological concepts covered in the lesson. • Emphasize the importance of understanding volcanic processes for risk assessment and mitigation. • Discuss the potential of using Minecraft Education Edition for other geography and science lessons. • Reinforce the dynamic nature of the earth.
Link for Challanges & CodeBlocks	minecraft volcano challenges and codes.docx
Game Link:	https://education.minecraft.net/en-us

Learning Scenario 3: Tinkercad Challenge: The Renewable Energy Rescue Mission	
Target Groups:	Students aged 15-18
Learning Objectives:	• Construct and manipulate 3D objects in a digital space. • Apply spatial reasoning and geometry to solve practical problems. • Experience basic design thinking and digital fabrication principles. • Develop digital literacy through CAD tools.
Materials & Resources:	• Computers with Internet access and Tinkercad accounts • "Escape Room Puzzle Sheet" • Projector for examples • Optional: 3D printer (for post-lesson activity)
Introduction (15 minutes):	• **Hook:** A digital vault must be cracked before time runs out! • **Context:** Introduce Tinkercad as a tool used by engineers, artists, and inventors. • **Relevance:** Explain CAD in architecture, robotics, prototyping. • **Demo:** Show how to combine shapes, align, group, resize.
Scenario Presentation (Step-by-Step)	
Teacher Tips and Strategies:	**1. Mission Briefing (5 min):** "Your team is locked in a virtual room. The only way out? Design five keys with specific properties (volume, angles, symmetry). Use Tinkercad to unlock each gate!" **2. Puzzle Sheet Distribution (2 min):** Each puzzle provides clues like: "Build a key with 3 shapes, total volume of 180 cm^3, and one right angle." **3. Tinkercad Refresher (5 min):** Quickly point out the most-used tools for this task. **4. Challenge Time (30–40 min):** Design, test with teacher's criteria, then move to next puzzle. Each puzzle is progressively harder. **5. Escape & Celebrate (5–10 min):** Teams that solve all 5 can "escape" and view other designs.
Learning Outcomes and Skills:	• Understanding of energy sources • Application of geography, engineering, and environmental science • Use of Tinkercad as a design and simulation tool • Teamwork and creative problem-solving
Feedback Collection (10 minutes):	• **Observe:** Spatial logic, collaboration, engagement. • **Puzzle Sheet Review:** Volume calculations, screenshot evidence. • **Exit Question:** "What real-world objects could you design with Tinkercad?"
Assessment & Evaluation:	• **Formative:** Observe students' collaboration and decision-making during the design process.

	• **Task Review:** Evaluate submitted designs and explanations for logical use of renewable systems. • **Optional Rubric Criteria:** Use of multiple energy sources, realism of layout, problem-solving, teamwork.
Conclusion (5 minutes):	• Celebrate the completed builds. • Discuss how such ideas relate to actual city infrastructure planning. • Encourage them to explore environmental engineering careers or urban design.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1KWdfaRCibyRhSofHd-Ph15nH4TGqn7HieVuuo-MvIuc/edit?usp=sharing
Game Link:	https://www.tinkercad.com/ https://education.minecraft.net/en-us

Learning Scenario 4: Cities: Skylines – Education Edition	
Target Groups:	Students aged 15-18
Learning Objectives:	Understand core concepts of **urban sustainability** and **resource management** Recognize interconnections between **infrastructure, environment, population health, and energy** Apply **critical thinking** in city development under resource constraints Explore **green technologies**, pollution control, traffic planning, and zoning Develop **soft skills**: collaboration, negotiation, scenario-based decision-making Introduce concepts of **circular economy**, **eco-innovation**, and **public policy design** Practice using **data visualization tools** for monitoring pollution, noise, traffic, and citizen satisfaction.
Materials & Resources:	• Computers with **Cities: Skylines** (Education Edition preferred) • Projector/Smartboard to present successful city designs or discuss failure cases • **Urban Planner Logbook** handout (for recording design ideas, decisions, and impact) • Printed **Challenge Briefing Sheets** Optional: Internet access for research on sustainable cities (e.g., Copenhagen, Singapore)

Introduction (15 minutes):	• **Hook:** Show real-life short video clips about urban smog, traffic congestion, green buildings, and city farming • **Context:** "You are urban planning teams hired to design a new eco-city for 100,000 citizens. Your job? Make it livable, clean, efficient, and green!" • **Relevance:** Connect simulation goals to UN SDGs (Sustainable Development Goals), especially #11: Sustainable Cities and Communities • **Tool Familiarization:** Quick demonstration of zoning, utilities, budgeting, traffic tools, and pollution monitoring in the game • **Team Formation:** Assign student roles: • **Sustainability Strategist** (focus on environment and energy) • **Infrastructure Planner** (roads, transport, zoning)

Scenario Presentation (Step-by-Step)

Mission Briefing (5 minutes):

- "Design a self-sufficient, environmentally friendly city with minimal pollution and traffic congestion within 60 in-game days."

Distribute Logbooks (2 minutes):

- Students use logbooks to plan zones, utilities, renewable energy strategies, etc.

Simulation Start (5 minutes):

- Teams begin laying out their cities. Set goals: low pollution, renewable energy %, citizen happiness.

Challenge Time (30–40 minutes):

- Students play the game in teams, making real-time decisions and adjusting to simulated problems (e.g., power shortages, traffic jams, trash overflow).

Crisis Scenarios (Optional, 10 minutes):

- Introduce surprise challenges: natural disasters, budget cuts, energy shortages. Teams must respond live.

City Showcase & Debrief (10–15 minutes):

Teams present city screenshots, explain major strategies, and share what worked or failed

Teacher Tips and Strategies:	**Promote reflection:** "What would be the real-world consequence of your city design?" **Guide inquiry:** "What causes traffic spikes? How does zoning affect pollution?" **Differentiate challenges:** • *Beginner task:* Survive 60 days with no blackouts • *Advanced challenge:* Reach 80% renewable energy use while maintaining citizen happiness > 75%
Learning Outcomes and Skills:	• **Environmental Literacy:** Understand emissions, waste cycles, green energy • **Systems Thinking:** See how different city parts interact (e.g., traffic ↔ pollution ↔ health) • **Digital Literacy:** Operate a complex simulation environment • **Civic & Global Competence:** Grasp real-world urban challenges through simulation • **Communication & Leadership:** Work in diverse roles within a team setting
Feedback Collection (10 minutes):	• **Group Reflection:** What trade-offs did your team make? • **Exit Ticket:** "If this were your real city, what would you prioritize next?"

Assessment & Evaluation:	• **Formative:** Observe team collaboration, decision-making, problem resolution • **Summative:** • Review team logbooks for strategy notes and adaptations • Screenshot analysis: Are traffic, energy, and citizen needs balanced? • Optional Rubric: Green energy % | Pollution levels | Happiness score | Resilience to crises
Conclusion (5 minutes):	• Celebrate sustainable city successes and "beautiful failures" • Discuss how gaming can simulate complex, real-world systems • Encourage students to explore careers in green tech, urban design, public policy • Offer links to real-life smart city projects and open urban data Portals
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1vUql53GMrpFpysQNLTwM-yktOOE628lBKIUHT8-ein8/edit?usp=sharing
Game Link:	https://www.paradoxinteractive.com/games/cities-skylines/about

Learning Scenario 5: MinecraftEDU Challenge: The Renewable Energy Rescue Mission	
Target Groups:	Students aged 15-18
Learning Objectives:	• Construct and manipulate 3D objects in a digital space. • Apply spatial reasoning and geometry to solve practical problems. • Experience basic design thinking and digital fabrication principles. • Develop digital literacy through CAD tools.
Materials & Resources:	• Computers with Internet access and Tinkercad accounts • "Escape Room Puzzle Sheet" • Projector for examples • Optional: 3D printer (for post-lesson activity)
Introduction (15 minutes):	• **Hook:** A digital vault must be cracked before time runs out! • **Context:** Introduce Tinkercad as a tool used by engineers, artists, and inventors. • **Relevance:** Explain CAD in architecture, robotics, prototyping. • **Demo:** Show how to combine shapes, align, group, resize.
Scenario Presentation (Step-by-Step)	
Teacher Tips and Strategies:	**1. Mission Briefing (5 min):** "Your team is locked in a virtual room. The only way out? Design five keys with specific properties (volume, angles, symmetry). Use Tinkercad to unlock each gate!" **2. Puzzle Sheet Distribution (2 min):** Each puzzle provides clues like: "Build a key with 3 shapes, total volume of 180 cm^3, and one right angle." **3. Tinkercad Refresher (5 min):** Quickly point out the most-used tools for this task. **4. Challenge Time (30–40 min):** Design, test with teacher's criteria, then move to next puzzle. Each puzzle is progressively harder. **5. Escape & Celebrate (5–10 min):** Teams that solve all 5 can "escape" and view other designs.
Learning Outcomes and Skills:	• Understanding of energy sources • Application of geography, engineering, and environmental science • Use of Minecraft as a design and simulation tool • Teamwork and creative problem-solving

Feedback Collection (10 minutes):	**Observe:** Spatial logic, collaboration, engagement. **Puzzle Sheet Review:** Volume calculations, screenshot evidence. **Exit Question:** "What real-world objects could you design with Tinkercad?"
Assessment & Evaluation:	• **Formative:** Observe students' collaboration and decision-making during the design process. • **Task Review:** Evaluate submitted designs and explanations for logical use of renewable systems. • **Optional Rubric Criteria:** Use of multiple energy sources, realism of layout, problem-solving, teamwork.
Conclusion (5 minutes):	• Celebrate the completed builds. • Discuss how such ideas relate to actual city infrastructure planning. • Encourage them to explore environmental engineering careers or urban design.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1KWdfaRCibyRhSofHd-Ph15nH4TGqn7HieVuuo-MvIuc/edit?usp=sharing
Game Link:	https://education.minecraft.net/en-us

Lesson Scenario 6: MBlock Mission: Satellite Launch Simulation	
Target Groups:	Students aged 15-18
Learning Objectives:	• Understand basic physics principles: gravity, thrust, velocity, and mass • Apply the scientific method through design and testing • Construct a stable spacecraft • Develop resilience and iterative problem-solving
Materials & Resources:	• Computers with mBlock • Mission Brief and Launch Logbook handouts • Projector or Smartboard for shared launches
Introduction (15 minutes):	• **Hook:** Show dramatic launch/failure clips • **Context:** Introduce the students as aerospace engineers • **Relevance:** Real-world space exploration needs similar skills • **Tool Familiarization:** Basic spacecraft parts, engines, staging, launch controls • **Team Formation:** Pairs or trios with roles like Pilot, Designer, Analyst

Scenario Presentation (Step-by-Step)	
1. **Mission Briefing (5 minutes):** "Design and launch a satellite to simulated world." 2. **Hand out Logbooks (2 minutes):** Include design sketches, test logs 3. **Tool Refresher (5 minutes):** Review important parts and concepts 4. **Launch Mission (5 minutes):** Teams begin building 5. **Challenge Time (30–40 minutes):** Test, revise, and launch missions 6. **Mission Debrief (10 minutes):** Discuss successful and failed missions	
Teacher Tips and Strategies:	• Emphasize trial and error • Pose guiding questions: "Where is your center of mass?" • Encourage analysis of failures
Learning Outcomes and Skills:	• Newtonian physics and forces • Engineering design • Critical thinking and iteration • Scientific documentation
Feedback Collection (10 minutes):	• Team reflections: "What failed and why?" • Exit Ticket: "If this was a real satellite, what would it observe?"
Assessment & Evaluation:	• **Formative:** Monitor progress, note teamwork and use of physics principles. • **Summative:** Review logbooks and final launch results. • **Optional Rubric Criteria:** Stability of design, scientific reasoning, problem-solving appr
Conclusion (5 minutes):	• Reflect on successes and crashes alike. • Connect mission goals to actual aerospace missions (e.g., ESA, NASA). • Inspire interest in STEM and physics-related careers.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1Z9eZIjKa3ZlT1yneHqxwwqn0QncEUmSXC19 **ovdje prilagoditi sve mblocku (mislim da se može samo izmijeniti sve iz kerbaledu chatgptom čisto zbog funkcionalnosti)
Game Link:	https://mblock.cc/

Multi ACT STDD Lesson Scenarios

Learning Scenario 1: Elements Quest: Exploring Chemistry Through Minecraft	
Target Groups:	• Students aged 14–18 years old • Secondary School Science / Chemistry classes • STEAM Clubs or Gamified Learning sessions
Learning Objectives:	By the end of the lesson, students will: • Understand chemical symbols and basic element identification • Demonstrate knowledge of chemical reactions (like water formation) • Apply critical thinking to solve oxidation-reduction (redox) challenges • Collaboratively rebuild a portion of the periodic table • Improve teamwork, problem-solving, and creativity skills within a gamified environment
Materials & Resources:	• Minecraft Education Edition or Java/Bedrock Edition (Creative Mode) • Pre-built Minecraft world (with Elements Quest zones) • Command Blocks, Redstone supplies • Computers/Tablets with Minecraft access • Printable worksheet (optional) for note-taking during gameplay • Teacher's Guide (this document!)
Introduction (15 minutes):	• Start by briefly explaining what STEAM means (Science, Technology, Engineering, Art, Math). • Talk about how games like Minecraft can model scientific concepts (simulation, experimentation, collaboration). • Quick Icebreaker: Ask students to name 5 elements they know in 30 seconds!
Scenario Presentation (Step-by-Step)	
Zone 1: Element Discovery Lab • Students answer chemistry questions by finding and identifying element tokens. • Players use anvils to rename papers with chemical symbols. • Correct answers open doors to next area. **Zone 2: Chemical Reactions Challenge** • Students must "combine" dropped Hydrogen and Oxygen papers. • Correct combination generates Water (spawned in-game). • Incorrect combinations trigger a "no reaction" message. **Zone 3: Redox Reaction Puzzle** • Students interact with levers representing oxidation states. • Correct lever settings unlock next puzzle.	

<table>
<tr><td colspan="2">

- Wrong choices summon "Chemical Guards" (skeleton mobs).

Zone 4: Periodic Table Assembly

- Students organize custom-named elements into item frames.
- Correct placement unlocks the final door.
- Completion triggers a "Congratulations" title screen.

</td></tr>
<tr><td>Teacher Tips and Strategies:</td><td>

- **Assign Roles:** Leader, Recorder, Navigator (if working in teams).
- **Guide, Don't Tell:** Offer hints if stuck but let students discover.
- **Time Checkpoints:** Announce time left every 10 minutes if using a timer.
- **Observe Skills:** Watch for communication, reasoning, and creativity.
- **Use a Visual Aid:** Post a simple real Periodic Table in class for reference!

</td></tr>
<tr><td>Learning Outcomes and Skills:</td><td>

- **Chemistry Knowledge:** Element symbols, chemical reactions, redox basics.
- **Problem-Solving:** Logical sequencing of steps to achieve goals.
- **Collaboration:** Working in teams to solve puzzles.
- **Creativity:** Thinking outside-the-box to approach challenges.
- **Technology Skills:** Applying digital skills in a learning environment.

</td></tr>
<tr><td>Feedback Collection (10 minutes):</td><td>

- **Quick class discussion:**
  "What was the hardest part?"
  "What did you learn about Chemistry that you didn't know before?"
- **Optional: Exit Ticket Form asking:**
  1 New Fact I Learned
  1 Challenge I Overcame
  1 Suggestion to Improve the Game

</td></tr>
<tr><td>Assessment & Evaluation:</td><td>

Criteria	**Method**
Chemistry Knowledge	In-game achievement (tokens collected, reactions completed)
Problem-Solving	Observation during gameplay
Teamwork	Peer feedback or teacher notes

</td></tr>
</table>

	Reflection — Short paragraph after game session ("What I learned") (Optionally, you could score these lightly if you want a grade.)
Conclusion (5 minutes):	• Celebrate completion with a group cheer or a Minecraft-themed prize (like a "Top Chemist" printable certificate). • Tie back to real-world applications: "Why is combining elements important in real chemistry?" "How can working together solve bigger scientific challenges?" Encourage students to brainstorm what other subjects could be gamified next time (Biology? Physics?).
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1J5cKq9p_BcwKycrItQ8tlEsTfgwGE6vKMeGsMZgXgKs/edit?tab=t.0
Game Link:	https://makecode.com/_MT87q1dH0hoX

Learning Scenario 2: Code & Create! – Building the Future of Science and Art	
Target Groups:	Ages 14-18 Ideal for high school students studying Art, Science, Technology, and Engineering.
Learning Objectives:	• Students will be able to create 3D models using Minecraft based on real-world scientific concepts. • Students will understand the importance of renewable energy and how to model it within a game using Code Blocks. • Students will engage with interactive art, developing a deeper appreciation for digital art creation. • Students will gain hands-on experience in coding within Minecraft's Code Blocks environment. Students will enhance their collaboration and problem-solving skills through teamwork challenges.
Materials & Resources:	• Minecraft Education Edition installed on student devices. • Access to Code Blocks within Minecraft. • Projector or interactive whiteboard for demonstrations. • Handouts with coding blocks and brief tutorials on how to use Minecraft Code Blocks. • Printed instructions for the various STEAM challenges. Markers and paper for sketching designs (optional).

Introduction (15 minutes):	• **Introduction to Minecraft Code Blocks:** Begin with a brief explanation of Minecraft's Code Blocks and how they will be used for coding in-game actions. • **Overview of the STEAM activity:** Introduce the students to the main STEAM challenges: Building a Solar System model using Code Blocks (Science). Designing and recreating biomes (Art & Science). Inventing renewable energy sources in Minecraft (Technology & Engineering). Creating an interactive art gallery (Art & Technology). • **Goal Setting:** Explain the overall goal of the activity: to create a Minecraft world that demonstrates scientific concepts and artistic design using coding as the tool for construction.

Scenario Presentation (Step-by-Step)

1. **Solar System Model Challenge (10 minutes):**
   Students will use Code Blocks to create planets, moons, and the sun.
   Emphasize the need for realistic proportions and orbits.
   Demonstrate how to code basic planet creation, orbit mechanics, and how to use various blocks (e.g., gold for the sun, grass for Earth).
2. **Biome Representation Challenge (10 minutes):**
   Students will select a biome (desert, forest, ocean, etc.) and recreate it in Minecraft using Code Blocks.
   Demonstrate how to generate terrain (sand for desert, trees for forest, etc.) and plant life (cactus, oak trees, etc.).
3. **Renewable Energy Source Challenge (10 minutes):**
   Students will choose either solar panels, wind turbines, or water wheels and use Code Blocks to automate the energy generation.
   Demonstrate how to create and activate energy sources in Minecraft, explaining the science behind each (e.g., how solar energy works during the day).
4. **Interactive Art Gallery (10 minutes):**
   Students will create an art gallery, integrating interactive art such as paintings, sculptures, and sounds.

Show students how to create blocks for paintings, sculptures, and use Code Blocks to trigger sounds or display descriptions when interacting with the art.

Teacher Tips and Strategies:	• **Encourage Creativity:** Remind students that while the challenges are based on scientific principles, there's plenty of room for creativity in how they design and build. • **Pairing Students:** Pair students with different skill sets—those who excel at coding with those who enjoy creative design. This will foster teamwork and ensure balanced participation. • **Use Examples:** Show sample worlds or snippets of code before the students start to provide a reference point. • **Break It Down:** For beginners, break down each challenge into small, manageable tasks. Encourage students to focus on one step at a time (e.g., first coding the sun, then adding planets, etc.). • **Monitor Progress:** Circulate around the room and assist students with debugging their code or overcoming challenges.
Learning Outcomes and Skills:	By the end of the lesson, students will have: • Problem-solving skills in coding and building within Minecraft. • An understanding of scientific principles (e.g., biomes, renewable energy, and the Solar System) applied in a virtual environment. • Developed artistic design skills through the creation of Minecraft worlds. • Enhanced collaboration and communication abilities by working in teams to solve coding challenges. • Gained a hands-on introduction to coding using visual blocks.
	• **Encourage Creativity:** Remind students that while the challenges are based on scientific principles, there's plenty of room for creativity in how they design and build. • **Pairing Students:** Pair students with different skill sets—those who excel at coding with those who enjoy creative design. This will foster teamwork and ensure balanced participation. • **Use Examples:** Show sample worlds or snippets of code before the students start to provide a reference point. • **Break It Down:** For beginners, break down each challenge into small, manageable tasks. Encourage students to focus on one step at a time (e.g., first coding the sun, then adding planets, etc.).

	• **Monitor Progress:** Circulate around the room and assist students with debugging their code or overcoming challenges.
Feedback Collection (10 minutes):	• **Feedback through Classroom Discussion:** Ask students to share what they learned and which challenges they found most interesting. Discuss any difficulties they encountered. • **Exit Ticket:** Hand out a short form with the following questions: What did you learn today about renewable energy or biomes? What was the hardest part of coding in Minecraft? What would you change or improve in your final project?
Assessment & Evaluation:	• **Formative Assessment:** Monitor students during each challenge. Take notes on their coding solutions and engagement with the task. • **Peer Review:** At the end of the activity, students will present their work. Peers will give feedback on creativity, accuracy (scientific and artistic), and teamwork. • **Final Evaluation:** Review each project (Solar System, Biome, Renewable Energy, and Art Gallery). Evaluate based on: Creativity and accuracy in designing the models and systems. Use of Code Blocks to automate and create functional elements. Collaboration and communication skills shown during the group activities.
Conclusion (5 minutes):	• **Wrap-up:** Summarize what students have learned about science, art, and coding in the Minecraft world. Discuss how these concepts relate to the real world. • **Reflection:** Ask students to reflect on how this experience could inspire future projects or careers in STEM fields. • **Closing Feedback:** Thank the students for their hard work and creativity throughout the lesson.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1qwOK1qSDdA-l_kcZymQhUW-Pvmugqj-JDTjNjVv-rvA/edit?tab=t.0#heading=h.sptd31r1uk78
Game Link:	https://makecode.com/_1mhDut5LK94A

Learning Scenario 3: Building Logic and Automation in a Virtual World	
Target Groups:	• Age Group: 14-18 years • Class Type: Technology/Computer Science students • Skill Level: Intermediate knowledge of programming or a basic understanding of logic and automation.
Learning Objectives:	By the end of this lesson, students will be able to: 1. Understand and use basic programming concepts like loops, conditionals, and event-driven programming. 2. Create automated systems using Minecraft Code Blocks to build and control objects. 3. Apply problem-solving skills to design functional solutions within the Minecraft world. 4. Demonstrate the ability to integrate coding with in-game mechanics (e.g., Redstone, moving platforms, AI interaction).
Materials & Resources:	• Minecraft Education Edition (or Java Edition with Code Blocks enabled) • Code Builder tool within Minecraft • Computer with Internet Access • Projector for teacher demonstration (optional) • Minecraft world template with coding challenges set up in advance • Pre-prepared tutorial handouts or step-by-step guides for students (if needed)
Introduction (15 minutes):	1. **Teacher Introduction (5 minutes):** Explain the purpose of using Minecraft for learning to code and how the game can be used as a platform for real-world logic and programming concepts. Discuss the significance of automation, problem-solving, and logic in technology today. 2. **Introduction to Minecraft Code Blocks (10 minutes):** Briefly introduce Minecraft's Code Builder tool. Walk through how students will use MakeCode (or Python) to write basic scripts for moving platforms, creating Redstone circuits, automating farms, etc. Provide a sample code block (like moving a block) and briefly explain its logic.

Scenario Presentation (Step-by-Step)

Step 1: Setup and Introduction to the Challenge (10 minutes)

- Present the first challenge: Create a Moving Platform using code blocks.
- Explain the challenge: Students need to create a platform that moves back and forth in the Minecraft world using code. They will work on this problem first using the provided world template.
- Provide a basic script to get them started.

Step 2: Individual Work on Challenges (25 minutes)

- Students work independently or in pairs, coding and testing their solutions to challenges:

  Challenge 1: Moving Platform
  Challenge 2: Redstone Circuit
  Challenge 3: Smart Door Challenge 4:
  Automated Farm Challenge 5: Combat
  Guard NPC

- Teacher walks around and provides guidance as needed.

Step 3: Final Project Creation (10 minutes)

- Students will use the knowledge from previous challenges to create a final project: something unique that integrates what they've learned (e.g., a fully automated city with moving platforms, doors, and guards).

Students share their final projects with the class at the end.

Teacher Tips and Strategies:	**1. Hands-On Guidance:** Ensure each student or pair of students is actively engaging with the Code Builder and running the code within Minecraft. It's important for them to see immediate results from their code. **2. Break Down the Challenges:** Encourage students to focus on one task at a time. If they're struggling, break the problem down into smaller chunks (e.g., first make the platform move, then add a trigger). **3. Collaborative Learning:** If students get stuck, encourage them to discuss with peers. Collaboration often sparks new ideas and solutions. **4. Challenge Extensions:** For faster learners, provide additional challenges like creating a working elevator, building a complex Redstone circuit, or automating a trap system.

Learning Outcomes and Skills:	1. **Technical Skills:** Basic understanding of MakeCode or Python coding syntax in the context of Minecraft. Ability to create automated systems using Minecraft's Redstone and command blocks. 2. **Problem-Solving Skills:** Develop critical thinking skills by creating efficient code to solve in-game challenges. Apply logic to design working systems and troubleshoot errors. 3. **Collaboration and Communication:** Work collaboratively on challenges, sharing ideas, and discussing solutions. 4. **Creativity:** Use creativity to design custom tech solutions in a game environment, such as building unique moving platforms, Redstone circuits, or automated farms.
Feedback Collection (10 minutes):	• Use a short, anonymous survey or quick reflection questions to collect feedback from students about the lesson. • Example feedback questions: 1. What was the most interesting challenge you completed today? 2. Which part of the activity did you find the most difficult? 3. How do you think the skills you learned today can be applied outside of Minecraft? • Alternatively, students can provide feedback verbally after their final projects are presented.
Assessment & Evaluation:	1. **Formative Assessment:** Observe students' participation during the challenge phases. Are they actively coding and troubleshooting? Check the functionality of the code for each challenge (e.g., does the platform move, does the farm automate properly?). 2. **Summative Assessment:** Evaluate students' final projects based on: ▪ Complexity (did they integrate multiple components like Redstone, AI, etc.?)

	▪ Functionality (does it work as intended?) ▪ Creativity (did they customize their project?) **3. Peer Review:** Allow students to present their final projects to the class. Encourage feedback from peers to highlight creative ideas and approaches.
Conclusion (5 minutes):	**1. Recap Key Learnings:** Summarize the key takeaways from the lesson: coding basics, logic implementation, and the integration of technology within Minecraft. **2. Encourage Further Exploration:** Suggest that students explore more advanced coding tasks in Minecraft and consider applying what they've learned to real-world programming projects. **3. Give Additional Resources:** Provide links to MakeCode tutorials, Minecraft modding resources, or other coding challenges they can continue exploring outside of class.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1IjWqiMjNHQG7Jj3rObqv9E7oWg48lS7yVIKqpvnwphY/edit?tab=t.0
Game Link:	https://makecode.com/_CRk6aDK9Fd0s

Learning Scenario 4: EcoTech Engineers: Building Renewable Energy Systems with Code	
Target Groups:	• Age: 14–18 years old • Subjects: Technology, Science, Engineering, Programming • Group Size: Pairs or teams of 3–4 students
Learning Objectives:	Students will: • Understand and simulate renewable energy systems (solar, wind, hydroelectric) using coding concepts. • Apply block-based programming to automate and control environmental simulations. • Collaborate and problem-solve in teams to build an efficient energy grid. • Develop critical thinking around sustainability and technology integration.

Materials & Resources:	• Minecraft Education Edition installed on computers/tablets. • Downloaded world file: GreenstoneValley_EcoTech.mcworld. • Access to MakeCode for Minecraft (block coding interface). • Projector or Smartboard for teacher instructions. • Student guide/handout (optional).
Introduction (15 minutes):	1. **Hook:** Start with a quick discussion — *"Where does our energy come from? Why do we need renewable energy?"* 2. **Explain the mission:** *"Today, you are EcoTech Engineers! Your job is to build and code working renewable energy systems to save Greenstone Valley!"* 3. Briefly show basic block coding concepts using Code Builder.

Scenario Presentation (Step-by-Step)

1. **Spawn in Greenstone Valley** — Students enter the world.
2. **Start the "buildSolar" challenge:**
   Code solar panels that activate during daytime using block coding.
3. **Move to the Wind Turbine site:**
   Code a turbine to spin when the "wind" command is triggered.
4. **Move to the Watermill site**:
   Detect flowing water and spin the waterwheel automatically.
5. **Final Challenge:** Connect all systems to build a Smart Grid:
   Solar panels (day)
   Wind turbines (storm)
   Water wheels (water detection)
   Create an optional central control hub using chat commands.

6. **Bonus Challenge (if time permits):** Add a battery storage coded system.

Teacher Tips and Strategies:	• **Demonstrate once:** Walk students through a basic coding example live. • **Use teamwork:** Assign coding, building, and designing roles inside teams. • **Encourage iteration:** Allow students to test, fail, and adjust their code. • **Check checkpoints:** After each challenge (solar, wind, water), check progress. • **Support coding syntax:** Help with logic blocks like If, Repeat, On Event. • **Time warnings:** Give 10-minutes-left and 5-minutes-left reminders.

Learning Outcomes and Skills:	Students will develop: • Computational thinking (event-driven programming, automation) • Scientific literacy (renewable energy systems, sustainability) • Collaboration and communication (working in teams) • Critical thinking and problem-solving (optimizing their grid) • Creativity and innovation (designing control centers and efficient layouts)
Feedback Collection (10 minutes):	• Quick Exit Ticket (Google Form or paper): "What energy system did you enjoy building most?" "What was the biggest challenge?" "One thing you learned about coding or science today?" • Short peer feedback in groups: *"One thing I liked about our team's system was..."*
Assessment & Evaluation:	**Criteria** — **Points** Solar panel coded and working — +5 Wind turbine coded and animated — +5 Watermill working with water detection — +5 Smart grid integration — +10 Bonus battery/control center — +5 Team collaboration and creativity — +5 Total Possible Points — 35 pts • 30–35 points: Master Engineer • 20–29 points: Junior Engineer • Below 20 points: Apprentice – needs more testing
Conclusion (5 minutes):	• Gather everyone and ask: *"What was the hardest part to code?"* *"How does coding relate to real-world renewable energy technology?"* • Celebrate all teams and their creative energy grids! • Preview next lesson: *Adding sensors and smart city elements.*
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1i99tDqzEmTSdp5C8Bvulf4iLjlT37z7oyTMR5n3Pim4/edit?usp=sharing
Game Link:	https://makecode.com/_7WHMjqbmofM5

Learning Scenario 5: Fractal Garden - Exploring Recursion and Creative Design with Minecraft	
Target Groups:	• Students aged 14–18 • STEAM (Science, Technology, Engineering, Art, Math) focus • Suitable for mixed-ability coding or creative classrooms
Learning Objectives:	• Understand recursion through visual coding. • Apply math concepts (angles, scaling, proportion). • Create artistic designs using algorithmic patterns. • Develop problem-solving and creative thinking skills.
Materials & Resources:	• Devices with Minecraft Education Edition installed • Access to Code Builder (Blocks) • Projector or large screen for teacher demonstration • Student notebooks for sketching patterns
Introduction (15 minutes):	• Quick discussion: ➔ "Have you seen trees that look like mini-trees on their branches? That's *recursion*!" • Show real-world examples of fractals (tree branches, snowflakes, broccoli). • Explain goal: ➔ "Today we'll code trees in Minecraft — trees that grow themselves, mathematically and artistically."
Scenario Presentation (Step-by-Step)	
1. Launch Minecraft & Set Up (5 min) Open a flat world. Open Code Builder (press C). 2. Build the Basic Script (10 min) Create on chat command "tree". Make function tree(length). Add code blocks as given. 3. Run and Test (5 min) Type /tree in the chat to watch the Agent build. See how small the tree is if length = 10. 4. Modify and Explore (15 min) Change length = 15 or 20. Change angles: make them 20° or 45°. Try changing 0.7 to 0.5 to see sharper, different fractals. Students can customize: use colored wool for branches or leaves.	
Teacher Tips and Strategies:	• Walk around during coding; help with function mistakes. • Encourage experimentation with angles and scaling.

	• For struggling students: ➔ Use pre-filled templates with missing parts to complete. • For advanced students: ➔ Challenge them to create double-layer fractals (tree on a tree).
Learning Outcomes and Skills:	• Computational Thinking: understanding recursion. • Math Skills: angles, scaling factors, geometric reasoning. • Artistic Expression: designing beautiful natural fractals. • Teamwork (optional): pair programming challenges. • Critical Thinking: problem-solving when trees don't "look right".
Feedback Collection (10 minutes):	• Short Exit Ticket: "What was the hardest part?" "What did you find most beautiful about your tree?" "One thing you'd change next time?" or • Quick class discussion: thumbs up / sideways / down for confidence.
Assessment & Evaluation:	• Basic Success: Agent builds a basic tree. • Intermediate Success: Students modify length/angle/scale creatively. • Advanced Success: Students create artistic or themed fractal gardens. Rubric: **Criteria** — **Beginning** — **Developing** — **Proficient** — **Advanced** Coding Completion — X — X — X — X Math Concepts Used — X — X — X — X Creativity & Design — X — X — X — X
Conclusion (5 minutes):	• Celebrate their fractal gardens — let students *tour* each other's creations. • Emphasize how coding + math can create beautiful art. • Challenge for next lesson: ➔ "Can you code a *snowflake fractal*? A *mountain fractal*?"
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1GP8FHlw-TOjmzvZxLxt7cCUPfhL9zdnwTU2QiKeaRF4/edit?tab=t.0
Game Link:	https://makecode.com/_MEL5Fk8ta48x

Learning Scenario 6: Cell Survival - Exploring Biology Through Minecraft	
Target Groups:	• Age: 14–18 years • Grade Level: High School
Learning Objectives:	By the end of the lesson, students will: 1. Understand and explain the basic structures and functions of a cell, including the nucleus, mitochondria, and cell membrane. 2. Use coding to automate and complete specific biology-related challenges in Minecraft. 3. Apply knowledge of cellular processes like DNA repair, virus defense, and energy production to solve challenges. 4. Improve critical thinking and collaboration skills through teamwork and coding problem-solving. 5. Gain hands-on experience with Minecraft Education Edition and block-based coding.
Materials & Resources:	• Minecraft Education Edition (with Chemistry Resource Pack activated) • Computers/tablets with Code Builder and MakeCode Blocks enabled • Internet access for research (optional) • Minecraft World/Map: Pre-built "Cell World" map (with Cytoplasm Zone, Nucleus, Membrane Gate, Mitochondria Zone) • Coding Instructions (provided via step-by-step tutorial) • Minecraft Biome Booklet (optional: printed booklets for students to follow and document progress)
Introduction (15 minutes):	1. Context: Begin the lesson by discussing basic cell biology concepts: the structure of a cell, what each part (nucleus, mitochondria, membrane) does, and why it's essential for survival. 2. Engagement: Show a quick video or demonstration of cells in action (like a YouTube video on DNA repair or cellular respiration). Then, ask students how they think these processes could be modeled in Minecraft. 3. Objective: Explain that today, they'll explore the inside of a virtual cell in Minecraft and complete challenges related to cell biology using code.
Scenario Presentation (Step-by-Step)	
1. Step 1: Introduce the Challenges Show students a visual of the Minecraft "Cell World", explaining each section:	

- Cytoplasm Zone: Safe space to build and modify cells.
- Nucleus: Where students will repair DNA.
- Membrane Gate: Where the agent defends the cell from viruses.
- Mitochondria Zone: Build the energy center of the cell.

2. Step 2: Assign Roles/Tasks

   Split the class into small groups. Assign each group one of the following challenges:

   - DNA Repair Challenge
   - Virus Defense Challenge
   - Nutrient Gathering Challenge
   - Mitochondria Energy Build Challenge

3. Step 3: Hands-On Coding

   Each group will use Minecraft Education Edition's Code Builder to input the code blocks provided. They will solve their specific challenge by programming an agent to perform tasks.

4. Step 4: Collaboration and Problem-Solving

   Students will work together to debug and improve their code. Encourage group discussions and collaboration to help them reach the solution.

5. Step 5: Review and Share

   Once the challenges are completed, have each group present their solution and explain the biology concept behind it (e.g., why DNA repair is necessary, how the mitochondria provide energy, etc.).

Teacher Tips and Strategies:	1. Start with Guidance: For beginners, guide students step-by-step through the coding process. Help them understand basic commands in Minecraft's Code Builder. 2. Encourage Group Work: Promote teamwork by asking students to collaborate on solving problems. Encourage them to help each other debug the code. 3. Be Prepared to Troubleshoot: Have some common solutions or tips ready in case students encounter technical issues or bugs in their code. 4. Use Visuals: Consider showing examples of completed challenges as visuals, so students can better understand the concept before diving into coding. 5. Real-World Links: Relate the tasks in Minecraft to real-world biology to help students see the connection between their in-game actions and the actual biological processes they are studying.
Learning Outcomes and Skills:	By the end of this lesson, students will:

	1. Knowledge: Have a deeper understanding of key biological processes such as DNA repair, virus defense, and cellular respiration. 2. Skills: Demonstrate proficiency in block-based coding using Minecraft's Code Builder and solve problems through logical thinking. 3. Collaboration: Improve teamwork and communication skills through group problem-solving and presentation. 4. Creativity and Innovation: Think creatively to develop solutions to challenges within the game, applying both scientific knowledge and coding techniques. 5. Critical Thinking: Develop critical thinking skills by interpreting biology concepts and implementing them in a Minecraft environment.
Feedback Collection (10 minutes):	1. Formative Feedback: Throughout the lesson, observe student engagement and provide immediate feedback on their progress. Ask questions to guide them in the right direction if needed. 2. Post-Lesson Feedback: After the activity, ask students to fill out a quick survey or questionnaire about: Their favorite part of the game. Which challenges they found the most difficult and why. What they learned about biology through the game. Example questions: "What biological process did you learn about today?" "How did coding help you understand the biology of a cell?"
Assessment & Evaluation:	1. Ongoing Assessment: Monitor students' progress as they work through the coding challenges. Offer formative feedback and help with troubleshooting. 2. Final Evaluation: Evaluate students based on: ▪ Completion of tasks (Did they complete the coding challenges and the biology-related objectives?). ▪ Understanding of biology concepts (Can they explain the biological process behind the game?).

	▪ Creativity in problem-solving (Did they come up with innovative solutions within the game?). 3. Peer Review: Have students evaluate each other's work by presenting their code and solutions to the class.
Conclusion (5 minutes):	1. Wrap-Up: Conclude the lesson by highlighting the key biological concepts learned during the game. Discuss how they can apply coding and game design to explore other scientific topics. 2. Reflection: Ask students to reflect on what they learned, both in terms of biology and coding, and how this knowledge could be applied to other areas of their studies or life. 3. Extension: For homework or further lessons, students could explore more advanced biology topics using Minecraft, such as genetic mutations, photosynthesis, or ecosystems.
Link for Challanges & CodeBlocks	https://docs.google.com/document/d/1l9bHbQP7HLYOfjcbvknR2idYhgBOUjUZjQWIKwNoFm8/edit?usp=sharing
Game Link:	https://makecode.com/_80cYhLRCK2vR

2nd Gymnasium of Nafpaktos Learning Scenarios

Learning Scenario 1: Eco-Rescue Mission: Saving the Planet with Renewable Energy	
Target Groups:	• **Age:** 12-15 years (Junior High School) • **Level:** Basic knowledge of environmental science (renewable vs. non-renewable energy) • **Group Size:** Small teams (2-3 students)
Learning Objectives:	• Understand the differences between renewable and non-renewable energy sources. • Apply engineering principles to design a sustainable energy solution in a game environment. • Develop problem-solving skills by troubleshooting energy distribution challenges. • Foster teamwork and creativity through collaborative game-based learning.
Materials&Resources:	• **Computers/Tablets** with internet access • **Scratch** (online or offline editor) • **Printed "Eco-Rescue Challenge Sheet"** (with tasks and reflection questions) • **Projector/Smartboard** for demonstrations • **Optional:** Physical materials for prototyping (paper, markers, craft supplies)
Introduction (15 minutes):	• **Hook:** Show a short clip of a dystopian city suffering from pollution due to fossil fuels. Ask: *"How can we save this city using renewable energy?"* • **Context:** Explain that students will become "Eco-Engineers" designing a game where players must power a city using solar, wind, and hydro energy. • **Tool Familiarization:** Quick demo of Scratch's motion/sensing blocks relevant to the game (e.g., "if-then" for energy activation).
Scenario Presentation (Step-by-Step)	
1. **Mission Briefing (5 minutes):** • *"A city's power grid is failing! Your team must build a Scratch game where players collect renewable energy sources (sun, wind, water) to restore power before time runs out."* 2. **Challenge Sheet Distribution (2 minutes):** • Tasks: 1. Design a sprite for the "energy collector" (e.g., a drone). 2. Code movement (arrow keys) to navigate the city.	

3. Create "energy sources" (sprites) that increase a "Power Meter" variable when touched. 4. Add a timer counting down from 60 seconds. 2. **Hands-On Work (40 minutes):** • **Phase 1:** Build the core mechanics (movement, energy collection). • **Phase 2:** Add obstacles (e.g., "pollution clouds" that slow the player). • **Phase 3 (Optional):** Customize with sound effects/art. 2. **Playtesting & Feedback (10 minutes):** • Teams swap games and give feedback using a "2 Stars & 1 Wish" format (2 things they liked, 1 suggestion).	
Teacher Tips and Strategies:	• **Differentiation:** Provide pre-made code snippets for struggling students; challenge advanced learners to add a "battery storage" mechanic. • **Debugging:** Encourage "pair programming" (one codes, one tests). • **Real-World Link:** Discuss how the game mechanics mirror real energy grids (e.g., intermittent solar/wind power).
Learning Outcomes and Skills:	• **Science:** Renewable energy concepts. • **Engineering:** Systems design and efficiency. • **Technology:** Scratch coding (events, variables). • **Art/Design:** Sprite and backdrop customization. • **Collaboration:** Team-based problem-solving.
Feedback Collection (10 minutes):	• **Exit Ticket:** • *"Name one renewable energy source and how it works in your game."* • *"What was the trickiest part of your design?"*
Assessment& Evaluation:	• **Formative:** Observe teamwork and debugging strategies. • **Summative:** Evaluate games using a rubric (e.g., functionality, creativity).
Conclusion (5 minutes):	• **Showcase:** 1-2 teams demo their games. • **Reflection:** *"How could this game teach others about sustainability?"* • **Extension:** Challenge students to prototype a real-world energy solution (e.g., a solar-powered model).
Link for Challenges & Code Blocks	https://docs.google.com/document/d/11peo74Sq1aSnxiivbE4llc-2hdh6ucTb7poCWwlFp40/edit?usp=sharing
Game Link:	• Scratch Online Editor: https://scratch.mit.edu/ • Tutorials: https://scratch.mit.edu/ideas

Learning Scenario 2: "Redstone Rides: Engineering a Physics-Based Theme Park in Minecraft" (STEAM Focus: Science, Technology, Engineering, Art, Math)	
Target Groups:	**Age:** 12–15 years old **Level:** Introductory physics concepts (forces, motion, energy) **Group Size:** Teams of 3–4 students
Learning Objectives:	Understand basic physics principles: potential/kinetic energy, force, motion, and simple machines Design and build working roller coasters, piston doors, elevators, and trap-based challenges Apply engineering logic using redstone circuits Practice creativity in theming and artistic design of rides and park areas Collaborate in teams to ideate, build, test, and improve working models Enhance problem-solving and iterative thinking through trial and error
Materials & Resources:	Computers with Minecraft Education Edition Projector or Smartboard Printed "Redstone Theme Park Blueprint Sheet" Access to MakeCode/Code Builder (optional coding) Graph paper and pencils for planning circuits
Introduction (15 minutes):	**Hook:** Show a 30-second Minecraft roller coaster video with cool redstone features (lights, sound, traps). Ask: "How do you think that was built?" **Context:** Explain that teams are now engineers hired to design and build the ultimate Minecraft theme park using redstone-powered systems. **Tool Overview:** Demonstrate redstone basics: power sources (lever, button, pressure plate), redstone dust, repeater, piston, dispenser, rails, command blocks. **Team Formation:** Assign roles: Redstone Engineer, Creative Designer, Ride Tester, Builder
Scenario Presentation (Step-by-Step)	
Mission Briefing (5 minutes): *"Redstone Realm, the world's newest Minecraft theme park, is hiring engineers to design thrilling attractions! From physics-defying coasters to clever redstone games—your park must impress!"* **Planning Phase (10 minutes):** Teams sketch 2–3 rides/attractions on the **Blueprint Sheet**, noting where redstone will be used (doors, carts, traps, lights). They must include: • A roller coaster ride • One interactive redstone game • One artistic or decorative feature using redstone (e.g. fireworks, music)	

Mini Tutorial (Optional - 5 minutes): Teacher shows how to make a piston door and how to use repeaters for timing. **Build Time (40–50 minutes):** • Teams build rides and interactive attractions • Use redstone wiring, rail systems, and logic blocks • Test features, fix bugs, iterate designs **Showcase & Tour (10–15 minutes):** • Teams walk each other through their rides • Demonstrate at least one interactive redstone feature live	
Teacher Tips and Strategies:	Encourage logic-based debugging: "Where is the redstone path broken?" Provide redstone kits (pre-built circuit examples) to struggling teams Allow partial automation: combine redstone with MakeCode if students are advanced Give creativity challenges: "Make a ride that tells a story" or "Include an elevator" Reinforce collaboration: team members should rotate roles during building phase
Learning Outcomes and Skills:	Physics: force, motion, energy conversion Engineering: circuits, mechanisms, systems thinking Technology: command blocks and redstone logic Art: visual design, theme, symmetry Math: symmetry, spatial reasoning, timing with ticks Collaboration and communication skills
Feedback Collection (10 minutes):	**Group Share:** "What was the most fun feature to build?" "What didn't work the first time?" **Exit Ticket:** "One physics concept I used today:" and "One redstone trick I learned:"
Assessment & Evaluation:	**Observation:** How students interact, solve problems, and test solutions **Blueprint Sheet:** Planning and intention reflected in final builds **Ride Functionality:** • Does the coaster run smoothly? • Do redstone systems activate as intended? • Is the experience safe and engaging? **Creativity:** Theming, use of decorative blocks, originality **Presentation:** Clarity, enthusiasm, ability to explain redstone logic
Conclusion (5 minutes):	Congratulate students on becoming "Redstone Engineers" Highlight how real engineers plan, test, and iterate using similar principles

	Suggest ways they could extend their project with sensors, timers, or story elements
Link for Challenges & Code Blocks	https://docs.google.com/document/d/17VaBkVIXb06HBD9aVw7oU9ZEcX_XvNqMoQy9CeD7b6Q/edit?usp=sharing

Learning Scenario 3: "Symmetry Studio: Create Digital Art with Math in Scratch"	
Target Groups:	**Target Groups:** • **Age:** 12–14 years old • **Level:** Basic understanding of coordinates, symmetry, and Scratch interface • **Group Size:** Teams of 2–3 students
Learning Objectives:	**Learning Objectives:** • Understand mathematical transformations: reflection, rotation, translation • Use Scratch to simulate and animate symmetric patterns • Design and code an interactive art piece that responds to user input • Combine artistic creativity with logical sequencing and coordinates • Improve algorithmic thinking, debugging, and teamwork skills
Materials & Resources:	Computers with Scratch (online: scratch.mit.edu) Projector or Smartboard Printed "Symmetry Studio Challenge Sheet" Optional: grid paper for sketching symmetry ideas Example Scratch projects for inspiration (e.g., spinning kaleidoscope, drawing tool)
Introduction (15 minutes):	**Hook:** Show a short animated example of a Scratch kaleidoscope or pattern art. Ask: "How do you think this was coded?" **Context:** Introduce Scratch as a creative coding tool that can simulate artistic symmetry using math logic. **Math Tie-in:** Discuss types of symmetry (vertical, horizontal, rotational) and coordinate transformations. **Scratch Overview:** Quick demo of motion blocks, pen extension, and broadcast messages. **Team Roles:** Suggest: Logic Coder, Art Designer, Tester
Scenario Presentation (Step-by-Step)	

Game Link:	https://education.minecraft.net/en-us

Mission Briefing (5 minutes):

"Welcome to Symmetry Studio! Today, you're going to combine math and art to create an interactive piece of digital symmetry art in Scratch. Think of it as math-powered creativity!"

Planning Phase (10 minutes):

Reflection

Rotation

Translation

- **Build & Code (40–50 minutes):**

Use the **Pen** extension to draw lines or shapes

- Add motion blocks with math-based positioning (mirror across x or y axis)
- Use loops to create repeated rotations or translations
- Include user interaction:
  - Press a key to change color or start a new drawing
  - Mouse movement to drag or position a shape

Art Gallery Showcase (10–15 minutes):

- Teams demo their projects
- Share which math transformation was used
- Optionally, post their project to the class studio on Scratch

Teacher Tips and Strategies:	Demonstrate just enough to get them started—let students discover artistic variations Provide sample code chunks (e.g., "how to reflect a sprite across y = 0") Encourage peer-to-peer debugging and remixing ideas Push students to explain their symmetry choices and how the code supports them Use grid overlays in Scratch backdrop for spatial reasoning
Learning Outcomes and Skills:	Math: symmetry, coordinate plane, transformations Technology: Scratch blocks, pen extension, interactivity Art & Design: use of color, repetition, and form Logical thinking: sequencing, loops, if-statements Collaboration & creative communication
Feedback Collection (10 minutes):	**Group Reflection:** "What was your favorite design element?" "What was tricky about drawing with code?" **Exit Ticket:** "I used _____symmetry in my project and learned _____."

Assessment& Evaluation:	**Scratch Project Review:** • Evidence of transformation logic in code • Functional interactivity • Artistic quality and symmetry **Planning Sheet Review:** Clear math-art idea and execution
• Teams sketch their pattern idea on grid paper (e.g., butterfly wings, mandala, spinning flower) • Choose which types of symmetry to include:	
	Presentation: Team clearly explains both design and math elements Optional Rubric Categories: • Code logic • Use of transformations • Creativity/artistic value • Team collaboration
Conclusion (5 minutes):	Celebrate all projects in the gallery Reinforce how math can enhance creative expression Encourage students to remix projects at home or explore more drawing tools in Scratch
Link for Challenges & Code Blocks	https://docs.google.com/document/d/1rtCpJfg3-brzFT-8_PGfuhnaUcco8sHz7BoRGDuNK3U/edit?usp=sharing
Game Link:	https://scratch.mit.edu

Learning Scenario 4 (Revised): "GeomeTricks: Painting with Shapes in Scratch"	
Target Groups:	**Age:** 12–15 years **Level:** Comfortable with basic geometric concepts; beginning to explore transformations and programming logic **Group Size:** 2–3 students
Learning Objectives:	Apply geometric principles to code and render accurate shapes in Scratch Understand and implement transformations: **rotation**, **reflection**, and **tessellation** Use mathematical reasoning to calculate angles and side lengths of regular polygons Integrate **nested loops** and **variables** in Scratch to design complex geometric art Explore the relationship between symmetry, pattern, and visual aesthetics Collaborate and reflect on the intersection of mathematics and artistic expression

Materials & Resources:	Computers with access to scratch.mit.edu Projector/Smartboard Printed **GeomeTricks Design & Planning Sheet** Optional: compass, protractor, graph paper for off-screen design
Introduction (15 minutes):	**Hook:** Show real-world examples of geometric art (Islamic tiling, digital fractals, Op Art). Ask: *"Can we recreate these patterns using math and code?"* **STEAM Context:** Explain how artists and architects use geometry, and how software can turn algorithms into visual works **Scratch Tools Demo:** Pen extension, angle math (360/n for polygons), use of variables, nested loops, and custom blocks **Team Formation:** Suggest: Code Architect, Visual Designer, QA Tester

Scenario Presentation (Step-by-Step)

Mission Briefing (5 minutes):

"You are digital geometry artists tasked with designing a complex piece of art composed entirely of coded shapes and mathematical transformations. Your canvas is Scratch. Your tools are polygons, loops, and your imagination."

Planning Phase (10–15 minutes):

- Use the **Design Sheet** to:
  - Calculate angles for at least two polygons (e.g., pentagon: 360° ÷ 5 = 72° turns)
  - Sketch how they will rotate, reflect, or tessellate
- Plan the structure of the artwork (central symmetry, radial layout, tiling effect)

Building and Coding (40–50 minutes):

- Draw polygons using loops and angle math
- Use variables to dynamically change side length or number of sides
- Add nested loops to create rotations or patterns
- Use if statements or keyboard inputs for interactivity (e.g., change color, reset canvas)
- Optional: Create **custom blocks** (functions) to draw different shapes or patterns

Art Showcase & Peer Review (10–15 minutes):

- Each group presents their Scratch artwork
- They must explain:
  - The geometry used (e.g., "this spiral is a rotated triangle with increasing side length")
  - Any challenges they encountered
  - Allow peer questions and constructive critique

Teacher Tips and Strategies:	Push students to move beyond simple shapes (e.g., using polygons as elements of a larger design) Offer a challenge bank: "Try drawing a tessellation of equilateral triangles and hexagons" Model how to break shape logic into reusable custom blocks (abstraction) Emphasize precision in mathematical reasoning when coding drawing algorithms Encourage creative problem-solving and remixing designs
Learning Outcomes and Skills:	**Geometry:** Polygon construction, angle sum, transformation, tessellation **Math Application:** Use of formulas and logical reasoning in design **Coding:** Scratch variables, loops, custom blocks, interactivity **Art & Design:** Composition, symmetry, repetition, balance **Team Collaboration:** Communication, planning, role sharing **Reflection:** Making explicit connections between code, math, and Aesthetics
Feedback Collection (10 minutes):	**Exit Ticket:** "The shape I'm most proud of coding is ____because ____." **Peer Reflection Prompt:** "How well does this design show balance or symmetry?"
Assessment & Evaluation:	**Formative Observation:** Teacher circulates and notes how well students: • Apply mathematical reasoning (e.g., calculating angles, understanding transformations) • Use Scratch tools (pen, loops, variables, custom blocks) appropriately • Collaborate and communicate ideas within their teams **Design & Code Review:** Teacher evaluates: • Accuracy of shapes (e.g., correct polygon construction and rotation logic) • Use of loops and variables to enhance design flexibility • Implementation of interactivity (e.g., reset button, color control) • Optional use of custom blocks to organize code **Artistic Quality & Creativity:** • Does the final design demonstrate symmetry, repetition, and visual balance? • Is there evidence of thoughtful color and layout choices? **Team Presentation:**

	During the showcase, teams briefly explain: • The math behind the shapes used (e.g., angle calculation) • The artistic choices in their composition Challenges they faced and how they overcame them
Conclusion (5 minutes):	Highlight the creativity and precision students demonstrated Emphasize the role of math in creative technologies and design careers Encourage students to remix or expand their artwork on their own Scratch accounts
Link for Challanges & Code Blocks	https://docs.google.com/document/d/16bMsj1Xrg7eEDSFuc1hPi5Du02pA59E5EDbsCkO2jt0/edit?usp=sharing
Game Link:	https://scratch.mit.edu/

Learning Scenario 5: "Save the Stream: Water Pollution Simulation with mBlock"	
Target Groups:	**Age:** 12–15 years **Level:** Basic science knowledge of water ecosystems and beginner familiarity with coding logic **Group Size:** 2–3 students
Learning Objectives:	Understand the causes, types, and effects of water pollution on ecosystems and human health Simulate water monitoring using **mBlock and virtual sensors** (or real if available: pH, turbidity, temperature) Apply coding skills to create a **water pollution detection program** using variables and conditional logic Design a **virtual or physical prototype** of a water-cleaning or monitoring solution Develop awareness of environmental challenges and foster solution- oriented thinking
Materials & Resources:	Computers with **mBlock** installed (https://mblock.makeblock.com) mBot or compatible robot (optional – if physical hardware is available) Projector/Smartboard Printed **Water Watchers Challenge Sheet** Internet access for pollution case studies or data

Introduction (15 minutes):	**Hook:** Show short clips or images of polluted rivers and marine environments. Ask: *"What would you build or program to help solve this problem?"* **Context:** Explain that students will become **eco-engineers** and use **coding + science** to detect and combat water pollution. **Science Connection:** Quick recap of water pollutants (chemical, plastic, biological), and how water is tested (pH, turbidity, temperature, conductivity) **Tech Prep:** Brief overview of **mBlock** interface and how it connects science logic with visual programming.

Scenario Presentation (Step-by-Step)

1. **Mission Briefing (5 minutes):**
   "You've joined the Water Watchers Task Force! Your mission is to develop a simulation or robot that detects pollution in local water and responds with action—either alerting the community or activating a cleanup system."
2. **Planning Phase (10–15 minutes):**
   - Use the **Water Watchers Challenge Sheet** to:
     - Research common water pollutants in lakes, rivers, or oceans
     - Sketch a design: pollution detection system, alert robot, or cleaning prototype
     - Choose inputs (e.g., sensor values or simulated readings) and outputs (e.g., LED alert, motor, message display)
2. **Programming Phase (40–50 minutes):**
   - In **mBlock**, students create a program that:
     - Simulates sensor data (or reads from real sensors)
     - Uses **if-else statements** to trigger alerts
     - Displays readings (e.g., "High turbidity detected! Warning!")
     - Optional: controls motors (e.g., moving robot to clean area or simulate response)
   - Integrate **broadcasts, variables**, and **loops**
   - Use graphic stage elements to simulate polluted water visually if no robot is used
2. **Demonstration & Sharing (10–15 minutes):**
   - Each team presents their prototype or simulation
     - Explain how it works, what pollution data it responds to, and how it could help in real life

Teacher Tips and Strategies:	No physical robot? Use **mBlock's device simulator** and stage sprites to mimic readings Offer data ranges for simulation (e.g., "pH below 6 or above 8 = problem") Challenge advanced students to use **multiple sensors** or a **dashboard** display Discuss real-world innovations like **autonomous trash skimmers** or **sensor buoys**
Learning Outcomes and Skills:	**Science Understanding:** Water quality indicators and pollution impact **Coding:** Conditional logic, sensor simulation, variable tracking **Engineering Thinking:** Designing solutions, input/output systems **Environmental Awareness:** Connecting technology with global sustainability goals **Collaboration & Communication:** Working in teams, presenting technical ideas
Feedback Collection (10 minutes):	**Reflection Questions:** • "What was the biggest challenge your program helped solve?" • "If you had more tools, how would you improve your design?" **Exit Ticket:** • "Today I learned _____about water pollution and _____ about programming."
Assessment & Evaluation:	**Observation:** • Are students applying science knowledge to real-world contexts? • Are they using mBlock blocks correctly (sensors, logic, output)? • Are teams communicating and debugging together? **Challenge Sheet Review:** • Completeness and clarity of initial plan and logic flow • Accuracy of simulation or hardware model **Prototype Evaluation:** • Functionality of pollution detection logic • Clarity and accuracy of alerts or feedback • Creativity and realism of the proposed solution **Presentation:** • Explanation of how the system works • Understanding of pollution issues and science behind their code • Ability to reflect on limitations and possible improvements

Conclusion (5 minutes):	Celebrate their innovative and environmentally conscious efforts Emphasize that small, code-driven ideas can lead to big changes Encourage continued exploration of environmental science and Robotics
Link for Challanges & Code Blocks	https://docs.google.com/document/d/1rvHdOW_dhG-LZmYx2oAmDTWl_o9IEnlDVFC0Ob8cLiU/edit?usp=sharing
Game Link:	https://mblock.makeblock.com/

Learning Scenario 6: "ReCraft City: Designing a Recycling System in Minecraft"	
Target Groups:	**Age:** 12–15 years **Level:** Intermediate Minecraft familiarity, basic knowledge of waste types and recycling **Group Size:** Teams of 2–3 students
Learning Objectives:	Understand types of waste (organic, plastic, metal, paper) and the importance of recycling Use Minecraft Education Edition to design and simulate a city-wide recycling system Apply engineering concepts to build **automated sorting stations**, **recycling centers**, and **waste reduction infrastructure** Use **Redstone logic**, **command blocks**, and **coding (MakeCode)** for automation Foster creativity by integrating sustainability with city design
Materials & Resources:	Computers with **Minecraft Education Edition** Access to **Code Builder (MakeCode)** Projector/Smartboard for demonstration Printed **ReCraft City Challenge Sheet** Reference materials on recycling processes (videos, posters, real city models)
Introduction (15 minutes):	**Hook:** Show pictures or a short video of a modern recycling plant or waste crisis (e.g., plastic island in the ocean). Ask: *"How can we use technology to fix this?"* **Context:** Introduce the challenge: *"Your team will build a recycling system in Minecraft that helps a virtual city manage waste better."* **Science Review:** Briefly go over waste types and how they're recycled **Tech Prep:** Show how Redstone can simulate mechanical sorting, and how Code Builder can automate object movement or detection

Scenario Presentation (Step-by-Step)	
Mission Briefing (5 minutes):	
• Education center or museum • Choose which automation features to use (Redstone gates, hoppers, MakeCode sorting logic) **Build & Automate (40–50 minutes):** • Build city zones and simulate waste drop-off and collection • Use Redstone + command blocks to create: • Color-coded bins with item filters • Minecart transport systems • Sorting machines or conveyors • Use MakeCode to automate messages (e.g., "Plastic bin full!") or scoring for correct recycling • Optional: Build an "Eco-School" where NPCs teach recycling facts **Showcase & Walkthrough (10–15 minutes):** • Teams guide peers through their systems • Highlight one key innovation or automation feature • Explain how it connects to real-world recycling problems	
Teacher Tips and Strategies:	Provide example builds or Redstone sorting models to help teams get started Encourage environmental messaging in signs, NPCs, or posters Guide students to test automation systems in steps: bin → sorter → output Provide extension options for advanced students: real-time tracking boards, energy-efficient systems
Learning Outcomes and Skills:	**Environmental Literacy:** Understanding waste categories and recycling processes **Engineering:** Building automated systems using Redstone and hoppers **Technology & Coding:** Using MakeCode to simulate logic (if-then, loops) **Design Thinking:** Creating a functional and visually clear system **Team Collaboration & Reflection:** Planning, testing, and explaining their work
Feedback Collection (10 minutes):	**Quick Share:** "What was the hardest system to build and why?" **Exit Ticket:** • "One real-world recycling idea I included was _____." • "One new way I used Redstone/Code Builder was _____."

Assessment & Evaluation:	**Formative Observation:** • Teams are engaged, collaborating, and testing systems • Students debug and revise automation based on testing **Blueprint Review:**
"ReCraft City has a waste crisis! Your team has been hired to build a smart, sustainable recycling system. You'll need to create sorting stations, educate citizens, and show how tech can help reduce waste." **Planning Phase (10–15 minutes):** • Teams brainstorm and sketch their **Recycling System Blueprint** • Define key locations: • Home waste bins • Collection & sorting center • Recycling plant	
	• Quality of planning, completeness of system diagram • Connection between waste types and bin logic **Build Evaluation:** • Sorting system works correctly (Redstone or MakeCode logic) • Recyclables go to correct locations • Visual clarity and creativity of build **Presentation:** • Clear explanation of how systems work • Environmental awareness demonstrated • Each student contributes something
Conclusion (5 minutes):	Congratulate students on building smart cities Discuss how real-world engineers and designers face similar challenges Encourage them to think about real recycling solutions in their schools and homes
Link for Challanges & Code Blocks	hhttps://docs.google.com/document/d/1pRvuH2onaLzA3ZKekCkg0rBZAERvH9hlO_dw8H9v5vA/edit?tab=t.0
Game Link:	https://education.minecraft.net/en-us

CONCLUSION

In conclusion, the development of 36 innovative and multidisciplinary STEAM learning scenarios marks a major step forward in modernizing education and making STEAM subjects more accessible, engaging, and relevant for students across Europe. By incorporating non-formal, game-based learning methods through platforms such as Scratch, mBlock, and Minecraft, this project brings fresh energy into classrooms and supports the growth of essential 21st-century skills such as critical thinking, creativity, collaboration, and problem-solving.

These learning scenarios not only raise students' interest and motivation but also equip teachers with new tools and strategies to deliver more interactive and meaningful STEAM education. Through this collective effort, we encourage a shift from traditional, examination-based instruction toward more experience-driven and student-centered learning environments.

This achievement would not have been possible without the outstanding cooperation, creativity, and commitment of our project partners. We extend our deepest gratitude to each partner country and organization for their remarkable contributions to the creati on of these learning scenarios.

We sincerely thank **Greece** for its strong leadership and contributions:

EK KAVALAS (E10199014 - EL) for guiding the project with vision and producing lesson scenarios that combine innovation and educational value.

2nd Gymnasium of Nafpaktos (E10222923 - EL) for its dedicated work in creating multidisciplinary scenarios that engage students through practical applications and creative thinking.

We warmly thank **Turkey** for its dynamic input and educational expertise:

University of Cukurova (E10208716 - TR) for contributing high-quality, academically rich lesson scenarios that bridge science and technology with game-based learning.

MULTI-ACT STD (E10283562 - TR) for bringing a strong focus on digital innovation and supporting the integration of interactive tools into STEAM instruction.

We greatly appreciate **Latvia** for its innovative and inclusive approach:

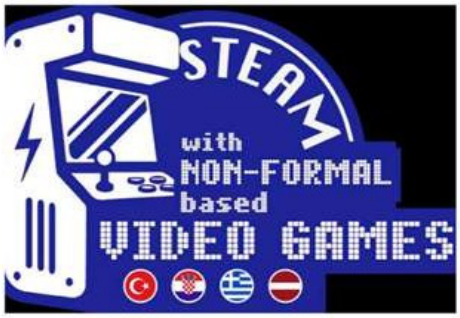
Rigas 64. vidusskola (E10150517 - LV) for designing engaging, student-centered lesson scenarios that promote active learning and cross-disciplinary connections.

We are also thankful to **Croatia** for its practical and impactful contributions:

Srednja skola Metkovic (E10040054 - HR) for developing hands-on learning scenarios that connect STEAM subjects with real-life challenges and foster deeper understanding.

Together, these organizations have built a valuable foundation for improving STEAM education through cross-sectoral collaboration and non-formal learning strategies. The scenarios created during this project will serve as lasting resources for teachers and students alike, helping shape future-ready learners across Europe.

Once again, thank you to all our partners—**Greece, Turkey, Latvia, and Croatia**—for your exceptional collaboration, shared passion for innovation, and dedication to educational excellence.



2024-1-EL01-KA220-SCH-000250956

Contributors:

EK KAVALAS / GREECE

Srednja skola Metkovic / CROATIA

2nd Gymnasium of Nafpaktos / GREECE

University of Cukurova / TÜRKİYE

Rigas 64. Vidusskola / LATVIA

MULTI-ACT STD / TÜRKİYE

Scan me!

Co-funded by the Erasmus+ Programme of the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the National Agency. Neither the European Union nor the National Agency can be held responsible for them.