# Prototyping a Game

**Prototyping a Game**

**Instructional Design and Gaming**

The fusion of instructional design with interactive platforms like **Unity** offers a transformative approach to learning. Instructional design, at its core, is about crafting educational experiences that are both effective and engaging. By utilizing models such as the ASSURE framework, educators can systematically design, develop, and evaluate interactive lessons tailored to their audience. Unity, a leading game engine, becomes an instrumental tool in this endeavor, allowing the creation of these lessons in a dynamic and immersive environment. This section aims to bridge the gap between instructional design principles and the practical application of Unity for educators. With a focus on the ASSURE model, educators will be guided on how to seamlessly integrate their instructional design knowledge into the Unity platform, even without a background in computer science or programming. The end goal is to empower educators to create interactive and impactful learning experiences that resonate with today's digital-native learners.

Before diving into Unity, it's essential to grasp the foundational principles of instructional design. The model, commonly used for lesson planning, will serve as our guiding framework for creating interactive lessons in Unity. This model will help you align your game design objectives, media selections, and assessments, ensuring that each game component fulfills a specific educational purpose.

**A: Analyze Learners**

* **Audience Analysis:** Understand the target learners for your Unity project. What are their interests, prior gaming experiences, and learning preferences?
  + **Further Exploration:** Read Chapter 3 of "." This section provides insights into learner analysis and its importance in instructional design.

**S: State Objectives**

* **Game Objectives:** Clearly define what the Unity game aims to teach or convey.
  + **Further Exploration:** Explore the SMART criteria for objective setting. Websites like can help you craft clear and measurable objectives.

**S: Select Methods, Media, and Materials**

* **Game Mechanics Selection:** Decide on the game mechanics that will best convey the subject matter.
  + **Further Exploration:** "Play to Learn" by Karl M. Kapp offers insights into various game mechanics and their educational implications.
* **Asset Gathering:** Collect or design the assets needed for the game.
  + **Further Exploration:** Websites like provide a plethora of resources. For design principles, "" by Robin Williams is a great start.

**U: Utilize Media and Materials**

* **Scene Organization:** Organize the game scenes in Unity in a logical sequence.
  + **Further Exploration:** Unity's official offers tutorials on scene organization and management.
* **Playtesting:** Test the game to ensure it is functional and engaging.
  + **Further Exploration:** Read "Games in Everyday Life: For Play" by Nathan Hulsey for a comprehensive guide.

**R: Require Learner Participation**

* **Interactivity:** Design game elements that require player interaction.
  + **Further Exploration:** "Interactive Design: An Introduction to the Theory & Application of User-centered Design" by Andy Pratt & Jason Nunes provides insights into crafting interactive experiences.

**E: Evaluate and Revise**

* **Peer Playtesting:** Have peers play the game and provide feedback.
  + **Further Exploration:**"" by Elbra-Ramsayand others offers strategies for effective feedback in educational settings.
* **Revision:** Refine the game based on feedback.
  + **Further Exploration:** The iterative design process is key. "" website delves into the importance of revision in design.

**Assessment:**

* Explore how the selected instructional design models can be adapted or incorporated into the game design process.
* Discuss the unique challenges and advantages of using instructional design methodologies in game development, especially for educational games.
* Elaborate on how these models can ensure effective learning outcomes while maintaining player engagement and interest.
* Discuss the interplay between instructional design and game design.

LLl**Learning Modalities and Game Design**

     Learning modalities, a concept rooted in the theory that individuals have preferred ways of receiving and processing information, often categorized into visual, auditory, reading/writing, and kinesthetic, play a significant role in the field of game design for learning. This approach suggests that educational games should cater to these diverse modalities to enhance learning outcomes. However, the application of in is not without controversy. Critics argue that the evidence supporting learning styles is weak and that designing games based on these styles may oversimplify complex learning processes. They contend that effective learning is more dynamic and less dependent on fixed modalities. Despite this, the integration of multimodal experiences in educational games—combining visual, auditory, and interactive elements—remains a popular approach. This strategy aims to engage a broader audience and acknowledges the multifaceted nature of learning, even as the debate over the validity and impact of learning modalities continues.

**"Watch the video tutorial onto prepare for your upcoming activities. Please complete this task promptly to engage fully with the materials and activities."**

**Assessment:**

* Do you think catering to different learning styles is beneficial in educational games, or do you agree with the critics who argue for a more dynamic approach to learning?
* How do you perceive the role of multimodal experiences in educational games, which combine visual, auditory, and interactive elements?

**3D Design**

This section is designed to guide you through creating your first level prototype in Unity using ProBuilder. After a brief introduction and learning phase, you will spend most of the week building a basic level, which will serve as a foundation for scripting and interactivity in a subsequent module.

ProBuilder is an innovative tool integrated within the Unity engine, designed to enhance and streamline the process of 3D modeling and level design directly in the Unity Editor. It stands out as a versatile tool for both beginners and seasoned developers, offering an intuitive interface that allows you to create, edit, and manipulate 3D objects without the need for external 3D modeling software. With ProBuilder, you can easily construct complex geometries, engage in rapid prototyping of game levels, and experiment with architectural designs. Its features include advanced modeling capabilities such as vertex, edge, and face manipulation, alongside UV mapping for detailed texturing. ProBuilder's seamless integration with Unity makes it particularly valuable for game developers, as it enables the immediate testing of 3D models within the game environment. This tool not only accelerates the workflow of game development but also opens up creative possibilities, making it an essential asset for anyone looking to harness the full potential of Unity in creating immersive 3D worlds and interactive experiences.

**Expected Outcomes:**

* Explore and familiarize yourself with the Unity ProBuilder interface to craft prototypes for educational games.

**Pre-Work Activity: Unity Interface Familiarization**

The purpose of this activity is to help you get acquainted with the Unity Editor's interface, ensuring you can navigate and utilize its various components efficiently when starting game prototyping with ProBuilder.

**Task:**

Unity Interface Overview:

* Open the Unity Editor.
* Allocate at least 30 minutes to explore the Unity interface on your own.
* Use Unity's Interactive Tutorials if available, which can be accessed via the Unity Hub or within the Unity Editor under the Window > Learn tab.

Key Area Identification:

* Identify and explore the following critical areas within the Unity interface:
  + Scene View: Understand how to navigate the 3D space, select, and manipulate objects.
  + Game View: Learn how to preview your game as it would appear during play.
  + Hierarchy Panel: Get to know how objects are organized and how to manage scene elements.
  + Project Panel: Look at how assets are stored and how to import new assets.
  + Inspector Panel: Familiarize yourself with object properties and how to change them.

Hands-On Task:

* Create an empty GameObject in the Hierarchy.
* In the Scene View, practice moving the GameObject around using the move tool.
* Rotate and scale the GameObject, observing the changes in both the Scene and Inspector panels.

**Documentation Reading:**

Read the 'Unity Interface' section of the Unity Manual available on the Unity Documentation to reinforce your understanding.

**Modeling and Animation with ProBuilder:**

**Please watch the following videos:**

[Intro to Unity Editor](https://www.youtube.com/watch?v=5UtptquK6gw)

[ProBuilder Part 1](https://www.youtube.com/watch?v=AWVQYwiViLE)

[ProBuilder Part 2](https://www.youtube.com/watch?v=ACagjHTtDL4)

[ProBuilder Part 3](https://www.youtube.com/watch?v=gfs3bgy-jFQ&amp;t=1s)

**Scripting in Unity Using C-Sharp**

This section is designed to introduce you to the essentials of scripting, which is the backbone of game development in Unity. We'll be using Visual Studio, a powerful and user-friendly code editor, to write and manage our scripts. Starting with the basics, we'll explore variables – the fundamental units of data storage. You'll learn how to use various data types and how variables interact within your game.

Next, we dive into the world of logic with 'if' statements, understanding how to make decisions in your code. This will lead us to loops – both 'for' and 'foreach' – which are critical for performing repetitive tasks efficiently. We'll see how these loops can work hand-in-hand with arrays, a key concept for managing multiple elements in an organized way. Lastly, we’ll cover methods, the building blocks of organized and reusable code, which allow us to perform specific tasks.

To learn C# specifically for Unity, it's important to choose resources that balance general C# concepts with their application in game development. Here are some recommended documents and websites for prework before diving into this module:

**Websites:**

1. **Unity Learn:**
   * Unity's official learning platform.
   * Provides courses specifically tailored to using C# in Unity.

**Video Tutorials:**

1. **Unity's Official YouTube Channel:**
   * Features tutorials and best practices.
   * Covers both beginner and advanced topics.

**Online Forums and Communities:**

1. **Unity Forums - Scripting Section:**
   * A great place to ask questions and learn from Unity developers worldwide.
2. **Reddit’s r/Unity3D:**
   * A community for Unity enthusiasts where you can discuss ideas and challenges.

**Please Watch the Following Videos:**

[Scripting in Unity Part 1](https://www.youtube.com/watch?v=oXtfZxsrHag&amp;t=1s)

[Scripting in Unity Part 2](https://www.youtube.com/watch?v=eI3UnYx5xoA)

[Scripting in Unity Part 3](https://www.youtube.com/watch?v=5C9brNJjchI&amp;t=1s)

[Scripting in Unity Part 4](https://www.youtube.com/watch?v=LiWJQVxDmKk)

[Scripting in Unity Part 5](https://www.youtube.com/watch?v=nwNKQsVKLhs)

[Scripting in Unity Part 6](https://www.youtube.com/watch?v=dYOb_9zvzhs)

[Scripting in Unity Part 7](https://www.youtube.com/watch?v=fVVWQxT41QM)

**Assessment. Unity Scripting Assignment: Grade Analyzer**

Objective: Create a Unity script that analyzes an array of grades and provides feedback about the class performance. You will define a set of grades, calculate the average grade, and use control structures to categorize the overall performance.

Assignment Details: In Unity, create a new scene and save it as "yourlastnameScene4", then create a new c# script in your script folder and name it " **QuizCalculator** " (Please use the camelcase style naming. Capitalize each word without spacing for writing the name of your script. However, when writing your variables, start with lowercase.; **for instance, gradeAverage.**

Here is the code that you need to complete. Remember, there are multiple ways to solve this problem. Aim to create the most efficient script possible!

------------------------------------------

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class QuizCalculator : MonoBehaviour

{

    // Declare an array to hold the quiz grades and a variable to hold the average grade

(Please write the code here)

    void Start()

    {

        // Initialize quiz grades with random values between 0 and 100 use for loop

(Please write the code here)

        // Calculate the sum of all grades using foreach loop

(Please write the code here)

        // Calculate the average grade

(Please write the code here)

        // Log the average grade to the console

  (Please write the code here)

        // Classify performance and log the letter grade to the console if and else if

(Please write the code here)

    }

}

-------------------------------------------------------

**Scripting in Unity Using C-Sharp Part2**

In this section, we will delve into user input, player movement, and collision handling in Unity, laying the foundation for creating dynamic and interactive games. You'll learn how to harness Unity's input system to translate players' actions—like keyboard presses and mouse clicks—into responsive on-screen movement. We'll explore how to use scripts to control characters or objects, adjusting their position, rotation, and scale based on user input, and implementing collision detection to create realistic interactions between game objects. Whether it's moving a character across a platform, steering a vehicle, navigating through obstacles, or interacting with the game environment, mastering player movement, input, and collision detection is a crucial step towards building engaging and immersive gameplay experiences. Additionally, we'll touch on Unity's UI system, understanding how to create and manage user interfaces that are both functional and aesthetically pleasing. 

**Please Watch the Following Videos:**

[User Input](https://www.youtube.com/watch?v=JTccQrCAAWk&amp;t=1s)

[Player Movement](https://www.youtube.com/watch?v=pfk8HOZCqQQ)

[Physic Part 1](https://www.youtube.com/watch?v=GtCe7po0-zE&amp;t=1s)

[Physic Part 2](https://www.youtube.com/watch?v=a1qKm5WtJVo&amp;t=1341s)

**Assessment 1: Player Health Reduction on Collision with Enemy**

**Objective:**Add a script to the player character in Unity so that when they collide with an enemy, they lose health. The player starts with 100 health points, losing 20 points with each collision. If the health points reach zero, the player is destroyed.

**Step-by-Step Pseudocode:**

1. Start with Player Health:
   * Define a health variable for the player and set it to 100.
2. Detect Collision:
   * Implement collision detection with an enemy object.
   * Use Unity's OnCollisionEnter or OnTriggerEnter methods.
3. Reduce Health on Collision:
   * When a collision with an enemy is detected, reduce the player's health by 20 points.
4. Check Health Status:
   * After reducing health, check if the player's health is less than or equal to zero.
5. Destroy Player if No Health:
   * If the player's health is zero or less, destroy the player object.
6. Optional - Display Health:
   * Optionally, include a way to visually display the player's current health (UI element, etc.).

**Here's a version of the Unity C# script for the player health system with key sections left blank for you to complete as an exercise: fill the blanks and create a script in unity and add it to your player. Test your game to see if it’s working correctly.**

public class YourLastNamePlayerHealth :MonoBehaviour

{

private int health = \_\_\_; // Students should fill in the initial health value

void OnCollisionEnter(Collision collision)

{

// Check if the collision object is an enemy

if (collision.gameObject.CompareTag("\_\_\_")) // Students should fill in the correct tag

{

// Reduce health

health -= \_\_\_; // Students should fill in the correct amount of health to deduct

// Check if health is zero or below

if (health <= \_\_\_) // Students should complete the condition

{

// Destroy the player object

Destroy(\_\_\_); // Students should fill in what needs to be destroyed

}

}

}

}

**Assessment 2: Play a Specific Sound on Collision**

public class YourLastNameEnemySoundEffect :MonoBehaviour

{

public AudioClipfunnySound; // Assign the specific funny sound clip

private AudioSourceaudioSource;

void Start()

{

// Get the AudioSource component attached to the enemy

audioSource = GetComponent<AudioSource>();

}

void OnCollisionEnter(Collision collision)

{

// Check if the collision object is the player

if (collision.gameObject.CompareTag("Player"))

{

// Play the funny sound clip

audioSource.PlayOneShot(funnySound);

}

}

}

**Instruction:**

1. **Attach the Script to Your Enemy GameObject:**
   * In Unity, select the enemy GameObject in your scene.
   * Drag and drop the **EnemySoundEffect** script onto the enemy GameObject in the Inspector to attach it.
2. **Add an AudioSource Component:**
   * With the enemy GameObject still selected, click on **Add Component** in the Inspector.
   * Search for **AudioSource** and add it to your enemy GameObject.
   * Uncheck the **Play On Awake** property of the AudioSource component if it is checked. This ensures the sound only plays upon collision, not when the game starts.
3. **Assign the Sound Clip:**
   * In the **EnemySoundEffect** script component, you'll see a field for the **funnySound**AudioClip.
   * Drag and drop your chosen sound clip from your project files into this field.
4. **Ensure Correct Tagging for Collision Detection:**
   * Verify that your player GameObject has the appropriate tag (e.g., "Player").
   * The script uses this tag to identify collisions specifically with the player.
5. **Testing the Scene:**
   * Run your scene and move the player to collide with the enemy.
   * Upon collision, the sound clip should play, indicating that the setup is working correctly.

**Publishing Your Game**

In this crucial phase of your game development journey, we're going to walk through the process of publishing your Unity game using WebGL. This powerful tool allows you to deploy and run your games on web browsers, making your creations easily accessible to a wide audience.

Here’s what you need to know to get started:

1. **Optimize Your Game for WebGL:** Before publishing, ensure your game is optimized for WebGL. This includes minimizing the use of resources, checking compatibility, and ensuring performance is up to par for a web environment.
2. **Build Settings Configuration:** Access the Build Settings in Unity and switch the platform to WebGL. This will enable you to configure various settings specific to the web platform.
3. **Building the Game:** With everything set up, build your game. Unity will compile all the necessary files and assets into a format that can be run in web browsers.
4. **Testing the Build:** It’s essential to test your WebGL build locally to ensure that everything works as expected. This can help you catch and rectify issues before going public.
5. **Choosing a Hosting Platform:** To share your game, you’ll need to host it online. Platforms like **Unity** offer great ways to publish and share your WebGL games.
6. **Uploading Your Game:** Watch the video Tutorial to upload the build files following the platform's guidelines.
7. **Sharing Your Game:** After your game is uploaded and published, you’ll receive a link that you can use to share your game with the world. You can embed this link into your Final website or share it on social media and forums.

**Call to Action:**

that covers each step-in detail. After watching, your task is to publish your own game using WebGL. Once it's live, take the sharing link provided by the hosting platform and integrate it into your website. This is a significant milestone, as it transitions your game from development to the hands of players. So, get your game ready, watch the video, and take that exciting step of publishing your creation for everyone to enjoy!

Read this online at <https://edtechbooks.org/designing_immersive_Learning_experience/prototyping_a_game>