Unity Piscine - Module03

Advanced inputs and 2D GUI

Summary: This document contains the subject matter for Module03 of the Unity Piscine.

Version: 2.1

Contents

Chapter I

Instructions

- If you have trouble installing the required tools for your project on the 42 computers, you may use a virtual machine. In this case, you must:
    - Install the virtual machine software on your computer.
    - Install the operating system of your choice.
    - Install the necessary tools for your project.
    - Ensure that you have enough space on your session to install all of it;
    - Have everything installed before the evaluation.
- Only this page will serve as reference. Do not rely on rumors.
- Carefully read the entire document before starting.
- Your exercises will be evaluated by your fellow piscine participants.
- This document is your reference. Do not blindly trust demos or example pictures, which may include unnecessary additions.
- Got a question? Ask the peer to your right. If not, try the one on your left.
- By Odin, by Thor! Use your brain!!!

`Intra shows the date and time when your repositories close. This also marks the beginning of the peer-evaluation period for that piscine day. The peer-evaluation lasts exactly 24 hours. After that, any missing evaluations will be scored as 0.`

Chapter II

Foreword

As you've seen earlier, you'll be reusing what you created in Module02 for today's Module03. So, make sure to import your assets into your Module03 project.
For this module, you will need at least two levels to demonstrate the transition between levels as well as the end of the game.

Chapter III

Exercise 00: A Simple Menu

	Exercise :
Exercise 00: A Simple Menu	
Turn-in directory : `unityModule03`	
Required elements : `"Menu" scene and anything useful`	
Forbidden functions : `None`	

Create a simple menu scene for our Tower Defense game. Let's keep it simple here. Just make a 'Play' (or 'Start', or whatever you prefer) button that launches the game, and a 'Quit' button that will close the application.

The 'Play' button must load the "map01" scene, which doesn't exist yet; but don't worry, you'll create it in the next exercise.

Find a nice-looking background and make an attractive layout!

Chapter IV

Exercise 01: Drag and Drop

	Exercise :
Exercise 01: Drag and Drop	
Turn-in directory : `unityModule03`	
Required elements : `"map01" scene, scenes and anything useful`	
Forbidden functions : `None`	

- `'Layers'` is your friend! It will help you manage drag and drop mechanics, and also prevent things like enemies or missiles from running underneath your turrets, for example.
- **`Cooldown:`** In video games, a cooldown is the amount of time an entity must wait before performing an action again. For our turret, it's the time you have to wait before it can fire again.

Objective: Create a turret selection bar and implement a drag-and-drop system for turret placement in your Tower Defense game.

Add a bar at the bottom of the screen that displays:

- The three basic turrets available with their:
  - Damage
  - Price (energy cost)
  - Cooldown
- Your base's HP
- Your energy reserve (needed to buy turrets)

Turret Placement:

- Create a grid of squares around the road where turrets can be placed.
- To buy a turret, the player must ‘drag and drop’ it from the bar onto a valid square.
- A turret can only be placed if:
    - The target square is empty
    - The player has enough energy
- If placement is valid:
    - Deduct the turret’s cost from the player’s energy reserve
    - You’ll need to implement this energy reserve system yourself:
        - Does it regenerate over time?
        - Do you earn energy by killing enemies?
        - Up to you!

Visual Feedback:
Provide visual cues to show whether a turret can be placed:

- For example, change the turret’s color if there isn’t enough energy.
- If the player can’t afford the turret, it shouldn’t be draggable.

`The color property is really useful - you can both GET and SET it. That means you can easily check or update a sprite’s color for visual feedback. Go explore the Unity docs!`

Chapter V

Exercise 02: Pause Menu

	Exercise :
Exercise 02: Pause Menu	
Turn-in directory : `unityModule03`	
Required elements : `scenes and anything relevant`	
Forbidden functions : `None`	

In this exercise, you'll add a 'pause menu' that appears when the player presses the `Esc` key.

To manage the pause state, take a look at the `Time.timeScale` property, it's your best ally here.

The Pause Menu Must Include:

- A 'Resume' button to return to the game.
- A 'Quit' button to exit the game.

Quit confirmation:

- If the player chooses to quit, display a 'confirmation menu' asking them to confirm their choice.
- If they confirm, return them to the 'main menu scene'.

Chapter VI

Exercise 03: I Am A Hero

	Exercise :
Exercise 03: I Am A Hero	
Turn-in directory : `unityModule03`	
Required elements : `"Score" scene, scenes and anything relevant`	
Forbidden functions : `None`	

In this exercise, you'll expand your game by implementing enemy wave control, a scoring and ranking system, and a complete end-of-game experience, including a dynamic score screen that appears whether the player wins or loses.

What to implement:

- Create waves of enemies that spawn at regular intervals.
  - Stop enemy spawning once a set number of enemies have been sent.
- Display the player's score and rank at the end of the game.
  - The score should be calculated in the `GameManager`.
- Create a 'ranking system' with at least 3 different ranks (e.g. F to S).
  - Feel free to get creative with the rank names!
  - Ranks should depend on the player's remaining life and energy at the end of the level.

End-of-Game Options:

- If the player loses, show a 'Replay' button.
- If the player defeats all enemy waves, offer a button to advance to the next level.

Progression and Final Screen:

- Create your final level (remember, you need at least 2 levels).
- Design a classy, satisfying end screen for players who complete your final level.
- Don't just throw them back to the title screen — give them a proper ending!

Chapter VII

Submission and Peer Evaluation

Submit your assignment in your `Git` repository as usual. Only the work inside your repository will be evaluated during the defense. Don't hesitate to double-check the names of your folders and files to make sure everything is correct.

You should not upload the entire Unity project to Git, as this can unnecessarily increase the size of your repository.

- Make sure Unity saves as many files as possible in text format rather than binary. In Unity, go to Edit > Project Settings > Editor. Under Asset Serialization, set it to Force Text.
- Ensure that the .gitignore file automatically generated by Unity is present.

The evaluation will take place on the computer of the learner or group being evaluated.