Lab 01 – Install and Explore GDevelop

Objectives

1. Install GDevelop
2. Explore GDevelop by solving some problems.

Introduction to GDevelop

Here is the GDevelop website. To use GDevelop, one can choose to run it online or install it into the machine. For this course, we will install GDevelop into the machine so that we can save and work on our project files locally.

If your Operating System is not Windows, you may check out the Installation instruction written here.

GDevelop Tutorials

There are many tutorials on GDevelop online. A huge compilation of written and video tutorials can be found here.

The following lists some of the tutorials that are sufficient for this lab. You can follow the listed tutorials sequentially:

1. Get Familiar With GDevelop
   - A video tutorial that covers the gist of the GDevelop`s general layout and briefly explains the individual component of the engine.

2. Interface
   - This tutorial covers all the editors in GDevelop and their functionalities.

3. Objects
   - This tutorial covers the objects that can be placed in a game.
   - For this lab, focus on **Sprite** and **Tiled Sprite.**

4. Events
   - This tutorial covers the events which are used to add game logic.
   - For this lab, it is too early to go deep into the types of events (e.g for each event, link events, etc). We will get to these in the later labs.

Feel free to consult your tutor anytime you have problems with any of the topics above.

Proceed to the Problems section once you have completed the tutorial above.

Problems

Now, challenge yourself with the following problems to explore GDevelop hands-on. For every problem, hints are given to assist you. If you are stuck, feel free to inform the instructor for further assistance.

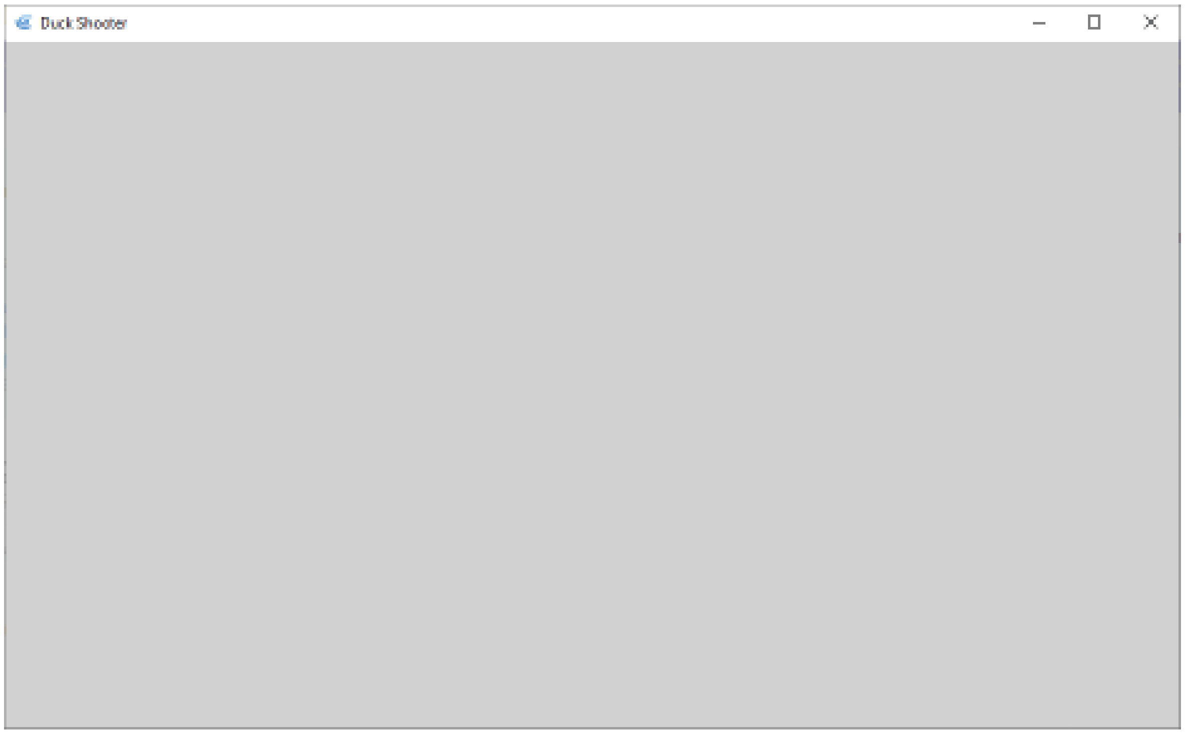
Before attempting these problems, create a folder named "Duck Shooter" in any path you desire. Then, copy and paste the Assets folder that comes with this lab into the Duck Shooter folder. Now, create a new GDevelop project and save its game. Json file into the Duck Shooter folder. You are now prepared to attempt the problems in the subsequent pages.

Problem 1: [Easy]

Set the width and height of the game window to **960** and **540** respectively.

Create a new scene. Then, edit the Scene properties to change the window title to "Duck Shooter". Finally, preview the project and you should get the following output.

Sample Output



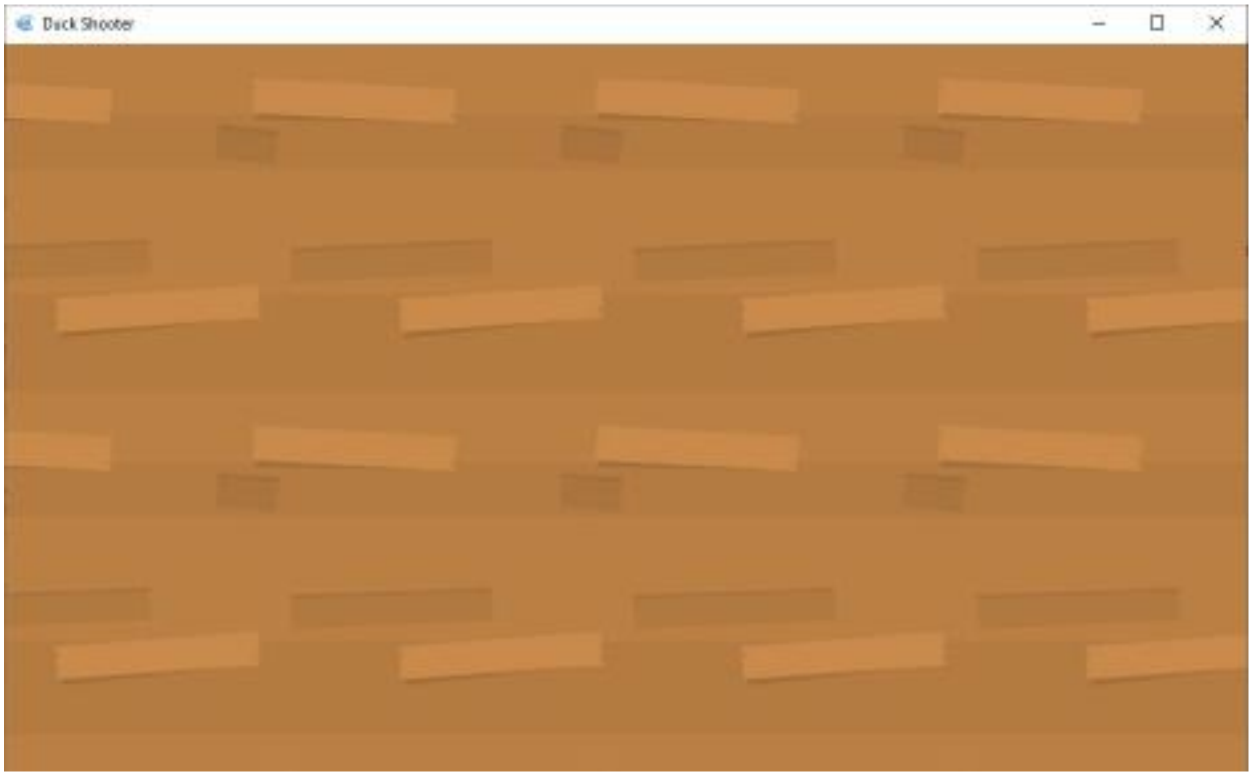
Hints:

1. Properties in Game Settings
2. Scene Properties

Problem 2: [Easy]

Create a Tiled Sprite and name it “BgWood”. Set its image to “Assets/Stall/bg_wood.png” and ensure its default width and height are set to the image’s width and height respective. Place only one BgWood object in the scene and make the necessary adjustments to get the following output upon previewing.

Sample Output



Hints

1. Tiled Sprite

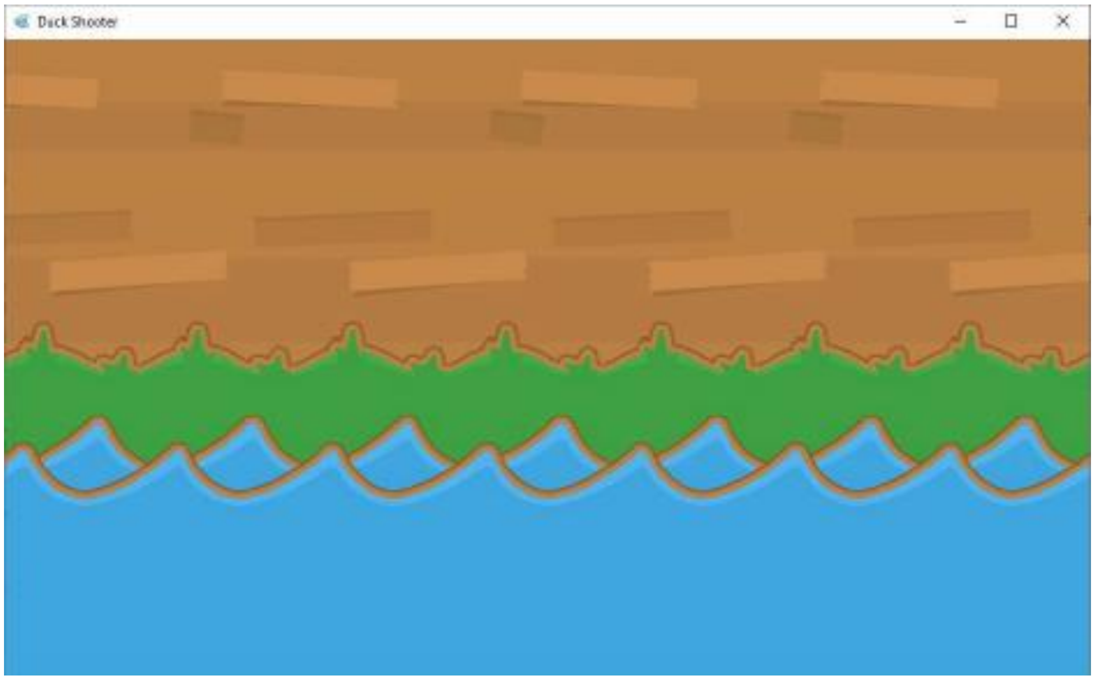
Problem 3: [Easy]

Create two new Tiled Sprites and named them “Grass2” and “Water2” respectively. For each tiled sprite, look for the images Assets/Stall/grass2.png and

Assets/Stall/water2.png and assign each image to its corresponding tiled sprite. Also, change the default width and height of each tiled sprite to its image width and height.

Create a new layer named “Platform”. Place these tiled sprites into the Platform layer and make the necessary adjustments such that the preview looks like the sample output below upon previewing.

Sample Output



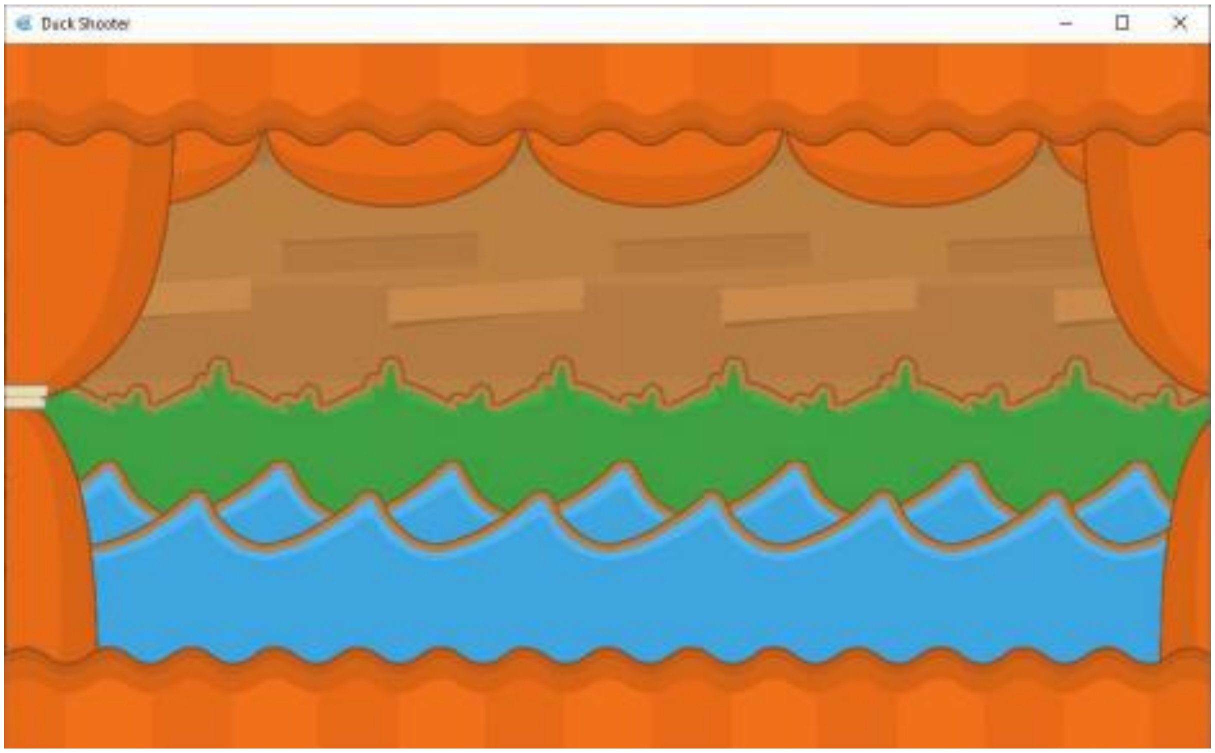
Hints

1. Layer

Problem 4: [Easy]

Check out the images in the folder Assets/Stall. Then, deduce what objects are needed to produce the output shown below. Place all the objects in a new layer named “Curtains”. Ensure that the Curtains layer is placed properly among all layers.

Sample Output



Hints

1. Sprite
2. Rotation
3. Layer Order
4. Z-Order

Problem 5: [Medium]

Create a new Sprite object and name it as "Crosshair". Add an animation and assign the image "assets/HUD/crosshair_outline_large.png" to it. Create a new layer named "Cursor" and make this layer the front-most layer. Then, Place one Crosshair object in the scene. In the Events editor, create an event such that the Crosshair's position always follows the Mouse's position at all times.

Sample Output

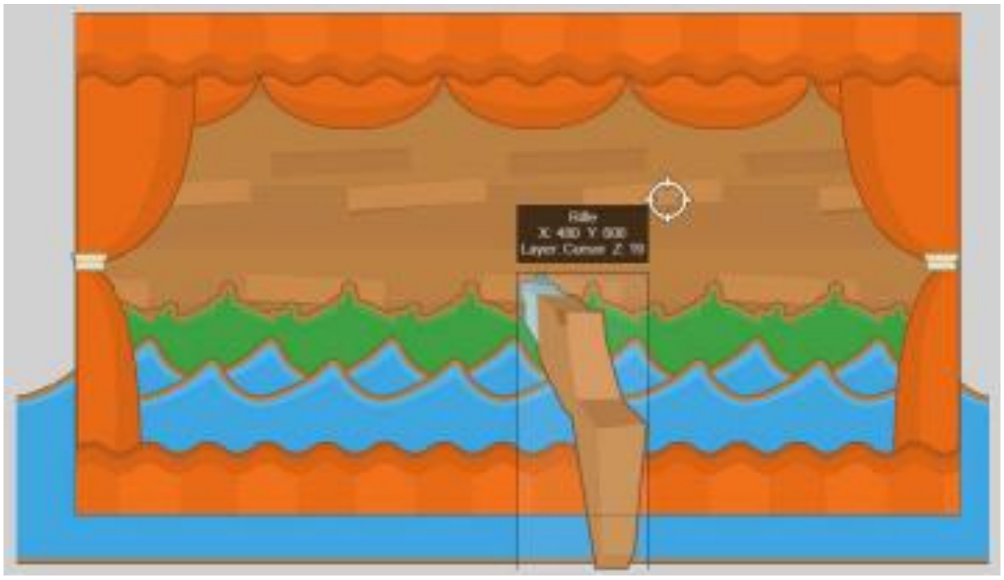
Check this video.

Hints

1. Event editor.
2. Zero condition, one action.
3. MouseX() and MouseY().

Problem 6: [Medium]

Create a Sprite object and name it “Rifle”. Set its image to “**Assets/Objects/rifle.png**”. Place **only one** Rifle object at the bottom of the scene like the figure below



The Rifle object **should only move left or right** with respect to the Crosshair position (which corresponds to the Mouse cursor position) to maintain a **fixed horizontal distance** between them. You may create a new layer or use existing layer to ensure the Rifle object appears in front of the curtains. Upon previewing, the output of your program should somewhat behave like the sample output.

Sample Output

Check this video.

Hints

1. Sprite
2. Layer
3. Edit Point

4. Zero condition, one action
5. X position

Acknowledgement

- The assets for the problems are obtained from Kenney's Shooting Gallery Pack.
- Original Lab Lesson prepared by Ban Kar Weng (William).
- Modified by ZulHisham Rozami.