Lab02 – Game Structure (Part 1)

Mission (should you choose to accept it)

1. Attempt the exercises on Formal Elements and discuss together with friends in class.
2. Explore GDevelop more by solving additional problems.

Practices

The following problems continues from Lab01 in which we work on the Duck Shooter game using GDevelop.

Attached with this document is a **Sounds** folder, which contains 3 audio files that will be used in this lab. Please place this folder into the **Assets** folder from the previous lab before attempting the problems below.

Problem 1: [Easy]

Let's give the player the ability to shoot. Create a Sprite object and name it **"`Shot`"**. Set its first animation to **"Assets/Objects/shot_blue_large.png"**. In the **Event Editor**, add an event such that when the **Left Mouse button** is pressed, an instance of `Shot` object is created in the scene and placed at the location of the centre of the `Crosshair` object. Upon previewing, the output should look like the Sample Output below.

Sample Output:

Check this video.

Hints:

1. Create Object (action)
2. `Crosshair.X()`, `Crosshair.Y()`
3. Edit Points

Problem 2: [Hard]

Create a Sprite object and name it **"`Bullet`"**. Set its 1st animation to **"Assets/HUD/icon_bullet_gold_short.png"** and name the animation as **"`gold`"**. Then, set its 2nd animation to **"Assets/HUD/icon_bullet_empty_short.png"** and name the animation as **"`empty`"**.

For the `Bullet` object, add an object variable named **"`id`"** and set it as a number primitive data type. Put zero (0) as the initial value for the `id` variable.

For the `Rifle` object, add **two** object variables named **"`numBullet`"** and **"`maxBullet`"** respectively. Set them as number primitive data type and put 5 as their initial values.

Place 5 instances of the `Bullet` objects at the bottom right of the scene and align them horizontally. You may put them in a separate layer to ensure that they are in front of the curtains as well as the `Rifle` object. Then, for the left-most `Bullet` object, change the value of the `id` variable to 1. Continue to change the `id` variable for the remaining `Bullet` objects from left to right, such that the right-most `Bullet` object would have the `id` variable set to 5.

With the above setup, you are ready to simulate the shooting action with bullet reduction. In the **Event Editor**, modify the events such that when the **Left Mouse Button** is pressed, the the `Rifle` object's `numBullet` variable is reduced by one (1), and that the right-most `Bullet` object with the "`gold`" animation is changed to the "`empty`" animation, indicating one button has been used. Upon previewing, the output should look like the Sample Output below.

Sample Output:

Check this video.

Hints:

1. Object Variables
2. Change the animation (action)
3. Trigger once when true (condition)

Problem 3: [Easy]

Add an action to reload the rifle when the **Right Mouse Button** is pressed. When the rifle is reloaded, all `Bullet` objects should be set to the "`gold`" animation, and the value of "`numBullet`" `Rifle` object variable should be updated correctly.

Sample Output:

Check this video.

Hints:

1. Trigger once when true (condition)

Problem 4: [Easy]

There are three sound files in the **Assets/Sounds** folder:

1. **gun-empty.wav**
2. **gun-reload.wav**
3. **gun-shot.mp3**

Your missions are as follows:

1. Play "**gun-empty.wav**" when the player shoots without ammo.
2. Play "**gun-reload.wav**" when the player reloads the rifle.
3. Play "**gun-shot.mp3**" when the player shoots with ammo.

Sample Output:

Check this video.

Hints:

1. `Audio`

Problem 5: [Easy]

Create a Sprite object and name it as **"`TargetColored`"**. Set its first animation to **"Assets/Objects/target_colored_outline.png"**. Place one `TargetColored` object into the scene somewhere on the left, like the screenshot below:

In the **Event Editor**, modify the events such that when the player shoots, a `Shot` object instance is created and placed into the scene **only if** the `Crosshair` hits the `TargetColored` object. Upon previewing, the output should look like the Sample Output below.

Sample Output:

Check this video.

Hints:

1. Point inside object (action)

Problem 6: [Medium]

Add a **Tween behaviour** to the `TargetColored` object. Then, add some tween events such that the `TargetColored` object alternates between moving to the right and to the left.

Sample Output:

Check this video.

Hints:

1. Behaviour
2. Tween behaviour
3. Trigger Once when true (condition)

Problem 7: [Medium]

Notice that when you shoot the moving `TargetColored` object, the Shot object does not move together with the `TargetColored` object. What can be done to ensure the Shot object moves together with the `TargetColored` object? Check out the Sample Output for what you are expected to accomplish.

Sample Output:

Check this video.

Hints:

1. Object Variables

Acknowledgement

The assets for the problems are obtained from Kenney's Shooting Gallery Pack.