Assignment 5: 2D Game Engine Project. GAME-10003-01 Ethan Muller

Game plan: I want to create pinball in Godot. Seems like a simple task. Pinball consists of the player using two paddels to keep the titular "pinballs" from falling down the middle of the screen. The player gains score by hitting objects above the paddles. These objects cause the pinball to bounce away from them. There are other objects that simply act as obstacles, not bouncing or providing score. The player has several balls in stock. The game ends when all balls are gone.

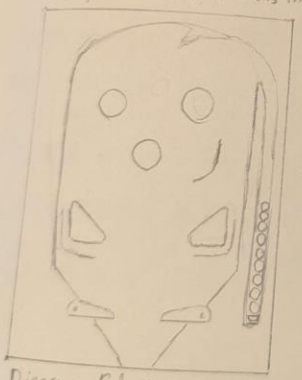


Diagram of typical game.

The main components to worry about are thus, the balls, the paddles and the bumpers.

After those components are completed I would focus on the layout, then score system, then the appearance.

The main components will be prefabs, of course. That way setting up the layout won't be Hell.

Pinballs:

The pinballs should be simple enough. I'll create a rigid body 2D node, with an ellipse shape and collision node connected. Creating multiple balls is more difficult. I'll have to either spawn the balls in ahead of time or create code that allows the balls to be spawned mid-game. The latter seems like a better solution, though I'll also have to code a counter, in order to prevent the player from spawning more than the maximum amount of balls.

I need to practice more with coding in Godot, as I'm not fully familiar with its terminology.

for the future. Impulse is different to Force. Impulse is momentary, force is Constant.)

Code suggested by prof: Vector2 p - Vector2 c = Direction.

Pinball = Rigid Body
Paddle = Animatable Body

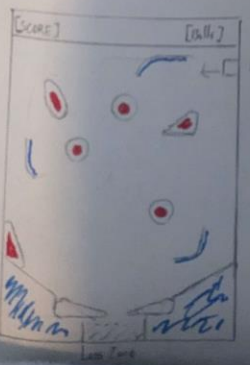
make animation player
make Animation track. Animation player node Make sure to add Action code.
Convert collision shape to paddle.
can keyframe bounciness.

Bumpers can be Static bodies with bounce

Turn off reset track in animation player.

The above are note from a discussion with the prof.

Since the objects should be more simple to layout. Creating Rigid/static/Animated bodies, then adding shapes and collision nodes parented to them, I'll try to layout what I want the Pinball game to be.



The score and amount of balls available to the player will be displayed at the top.

The paddles are placed above the pinball loss zone. The player loses 1 of ten balls when they fall into said zone.

The bumpers are layed to create chaos by bumping away the ball, while adding to the score.

The ramps are static, non-bouncy objects made to divert the balls.

The ball is fired at speed from the top right. There is no "win condition", the game going as long as the player has balls. The player goal is again the highest score possible.

(for the future. Impulse is different to force. Impulse is momentary, force is Constant.)

Code suggested by prof: Vector2 p - Vector2 c = Direction

Pinball = Rigid Body
Paddle = Animatable Body
make animation player
make Animation track. Animation player node. Make sure to add Activation code.
Convex collision shape to paddle.
can keyframe bounciness.

Bumpers can be Static bodies with bounce

Turn off reset track in animation player.

The above are note from a discussion with the prof.

Since the objects should be more simple to layout. Creating Rigid/static/Animated bodies, then adding shapes and collision nodes parented to them, I'll try to layout what I want the pinball game to be.

The score and amount of balls available to the player will be displayed at the top.

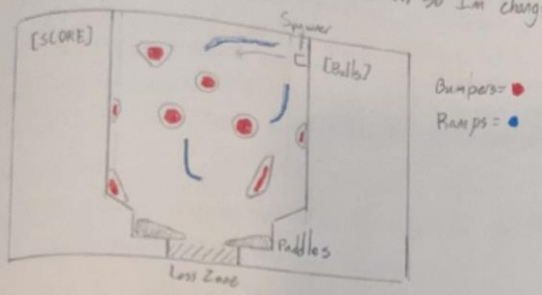
The paddles are placed above the pinball loss zone. The player loses 1 of ten balls when they fall into said zone.

The bumpers are layed to create chaos by bumping away the balls, while adding to the score.

The ramps are static, non-bouncy objects made to divert the balls.

The ball is fired at speed from the top right. There is no "win condition", the game going as long as the player has balls. The player goal is aquire the highest score possible.

The canvas is difficult to mess with so I'm changing the layout.



The ramps are difficult to make at the moment. I can't figure out how to shape them. I'm going to delay them and try to set up the ball spawner mechanic. I'm going to attatch a script to the ball. I'll make it so that when I want the ball to spawn it will be moving to the left at speed. I should be able to use an impulse command for that. Now I just need to figure out how to spawn these at the location I want.

Just had a good idea for the ramps. Using several rectangles to make a large prefab and then scaling them down I can have ramps.

I'm struggling with a bug I found concerning the wedge shaped bumpers. The bumpers use three rectangular collision boxes, but the ball is getting glitched out by the corners.
I've added circle colliders to the corners, they somewhat lessen the glitchiness. However, there's still problems. I'll try to move the rectangle colliders further into the wedge, just in case the ball is snagging on a small edge.

It seems I've traded one bug for another. The ball is stuck. I'm going to change the edge collision circles to rectangles.

Nope. That's worse.
Let's try one-way collision.
Nope.
changing the scale seems to have fixed the issue. Strange.

I've decided that the detection of the ball is more important than spawning the balls presently. I'll create a detection box below the hole and test to see if the box works. As in a demonstration created detection boxes using Area2D nodes and collision boxes. Luckily, I only needed one. With help from the professor I was finally able to understand both how to make detection boxes and spawn objects. These both needed exports which, when I was able to practice them, I better understood. Exports allow certain sections of a scripts code to be accessed and connected to nodes in the editor. I now need to spawn balls whenever a ball enters the detection box, which I lovingly refer to as "the Loss Box"

I'm thinking that if I place the initial spawning code into the detection boxes section of the level code it should spawn a new ball instantly.

The code would look like this:

```
private void LossBox_Body Entered(Node2D body)
{
    RigidBody2D NewBall = BallScene.Instantiate<RigidBody2D>();
    NewBall.Position = Marker.Position;
    BallCollection.AddChild(NewBall);
    NewBall.ApplyCentralImpulse(Vec);
    body.QueueFree();
}
```

This is a Node specifically made to be added

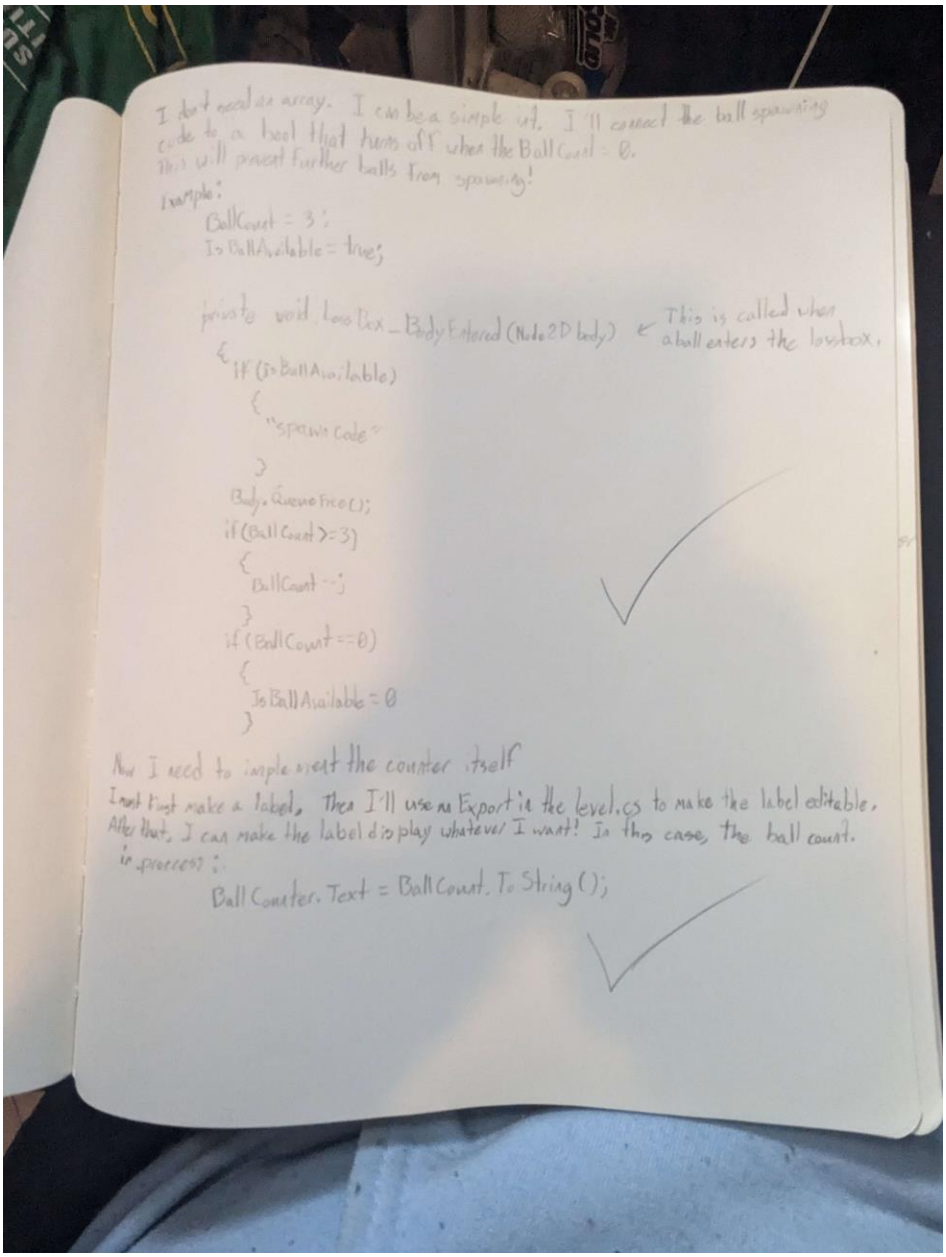
A Node all Balls are added to

Vec is a Vector2D that is exported.

My next job is to create a counter for the balls. This will keep the player from infinitely playing. I should be able to create a an int array between 0 and 2. Starting at 2 the balls will decrease. When a ball falls into the LossBox and the int is 0, the game will be over.

```
int BallCount = new int[3];

private void LossBox_Body Entered(Node2D Body)
{
    if (BallCount >= 0)
    {
        BallCount += 1;
    }
    if (BallCount = 0)
    {
        "Display game over"
    }
```



I don't need an array. I can be a simple int. I'll connect the ball spawning code to a bool that turns off when the BallCount = 0.
This will prevent further balls from spawning!

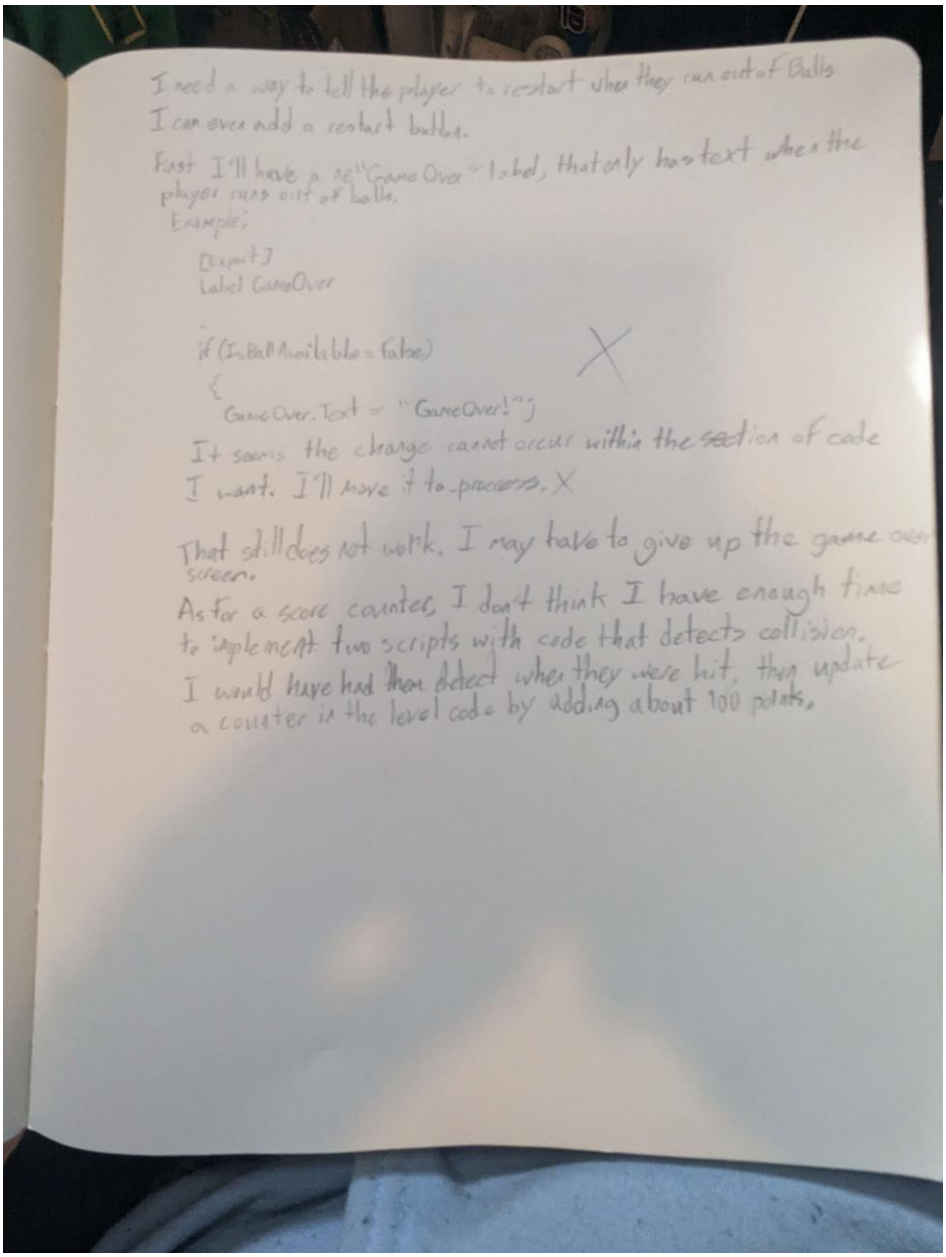
Example:

```
BallCount = 3;
IsBallAvailable = true;

private void LossBox_BodyEntered(Node2D body)  ← This is called when a ball enters the lossbox.
{
  if (IsBallAvailable)
    {
      "spawn code"
    }
  Body.QueueFree();
  if (BallCount >= 3)
    {
      BallCount--;
    }
  if (BallCount == 0)
    {
      IsBallAvailable = 0
    }
```

Now I need to implement the counter itself
I must first make a label. Then I'll use an Export in the level.cs to make the label editable. After that, I can make the label display whatever I want! In this case, the ball count. in process:

```
BallCounter.Text = BallCount.ToString();
```



I need a way to tell the player to restart when they run out of balls.
I can even add a restart button.
First I'll have a "Game Over" label, that only has text when the player runs out of balls.
Example:

```
[Output]
Label GameOver

if (isBallAvailable = false)
{
  GameOver.Text = "GameOver!";
```

It seems the change cannot occur within the section of code I want. I'll move it to process. X

That still does not work. I may have to give up the game over screen.
As for a score counter, I don't think I have enough time to implement two scripts with code that detects collision. I would have had them detect when they were hit, then update a counter in the level code by adding about 100 points.