Project 2 GDD

Firas el Jerdy

CSC 375 Spring 2019

Abstract:

I implemented the behaviours in the game and then decided to work extensively on the design aspect, because I never really made any fun games. So I made the project somewhat consistent graphically, and added the elements that make it interactive. Like screen shaking, particle effects and game mechanics. But due to unforeseen circumstances I couldn't continue shaping the game i wanted and thus stopped short from it. So you may notice it needs more sounds in some places, more UI and some mechanics.

Game Design:

Story: In a small town of Gothic-Vania, a notorious pack of ruthless monsters decided to pillage it and enslave its fine people. BUT, they didn't know that Catalina the Treant slayer was in town for the day to rest, before heading out the next morning to continue her journey as a bounty hunter.

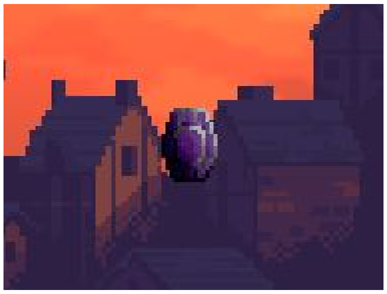
Mechanics:

Movement: Catalina is moved either LEFT, RIGHT or UP by respectively pushing the A, D and W keys on the keyboard. Velocity and friction influence the movements.

Shooting: Catalina shoots her thunder light machine gun with such kick, that it causes the ground to shake and instill fear in her enemies. The weapon is shot with the Left-Click mouse button.

Power-ups:

Catalina can collect *two* distinct power-ups in the world to tilt the odds of the battle in her favor. Power-ups, once collected, can be used in battle by pressing the Right-Click on the mouse button. The power-ups are said to be lost souls wandering around, ready to defend the village in anyway they can. A preview of the power-up can be viewed in the window on the top-right of the screen.



Power-up 1: The Boompocalypse

This power-up is a bomb, emitting a scent so beautiful, no one could resist the temptation of taking a whiff of it. Enemies follow it for a quick smell but only to find a blast that shreds their lost souls.

Power-up 2: The Fat-Sweaty man

Once activated, an atrocious smell is released from Catalina, that makes her enemies flee in terror. Some people say that the smell is so strong it makes the mind goes crazy. Take advantage of their weakness to take enemies out for a short period of time.

Mario style jumping:

Legend says that the Gothic-Vania raiders had no remorse... and also bouncy heads. Catalina can utilize this to her advantage.

Jumping on their heads gives Catalina a slight upward boost, great for escaping from her enemies

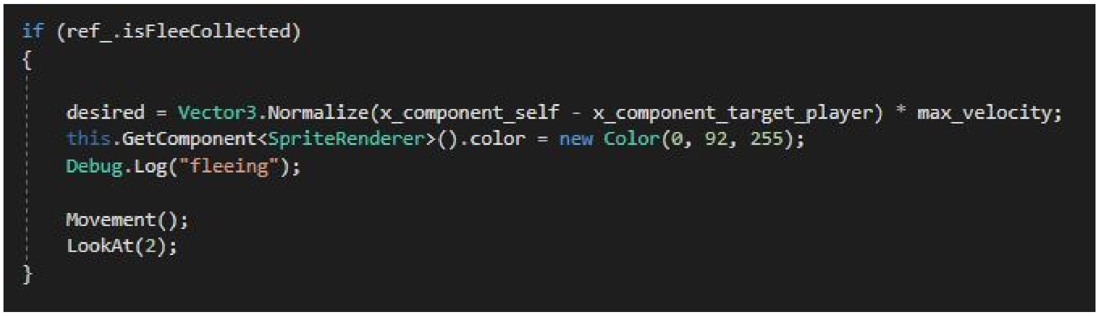
Enemies:

The village raiders are forged in the heart of the deep, dark woods. Ready to inflict harm to all kind people. The enemies are called the T.R.E.A.N.T (Tree's Running and Enslaving All Nice Townspeople). Treant's are ruthless killers capable of running in packs to tire their preys - like wolves. They may look tiny and cute, but their body masses are so enormous, that once a collision occurs, instant death befalls to anyone on the receiving end. So be careful. Treants are also known to have keen senses on their surroundings, capable of avoid all collisions. Nothing can stop them, not even morning breath. Platforms don't stop them.

Technical Remarks:

NPC's:

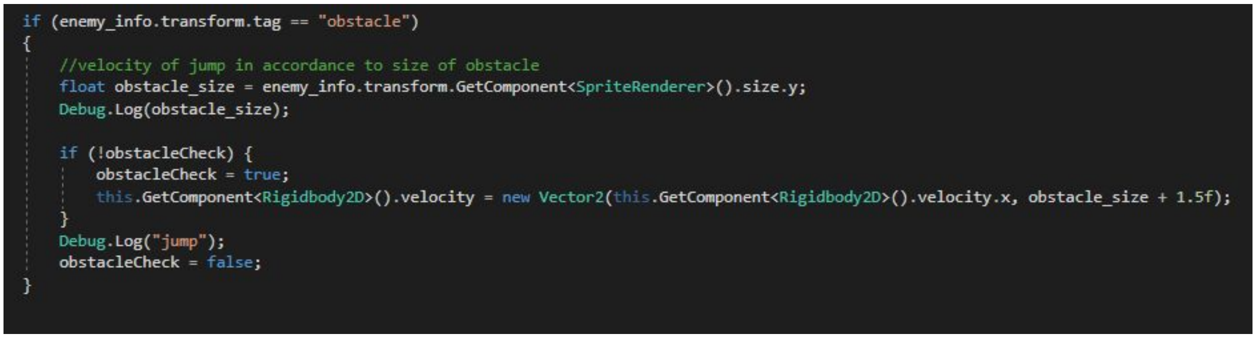
All NPCs are just one type of enemy - Treant. All enemy Treant's have a script called *TreantEnemyScript*.
The script handles the *Flee*, *Arrive*,*Collision Avoidance* and *seek* behaviours. Treant arrives to the player, when he touches him, instantly kills him. When the Bomb power-up is enabled, they switch their attention to that instead of the player. When the Flee power-up is used, Treant runs in the opposite direction of the player.



```
if (ref_.isFleeCollected)
{

    desired = Vector3.Normalize(x_component_self - x_component_target_player) * max_velocity;
    this.GetComponent<SpriteRenderer>().color = new Color(0, 92, 255);
    Debug.Log("fleeing");

    Movement();
    LookAt(2);
}
```

This is done by just switching the player position with the treant position in this equation.

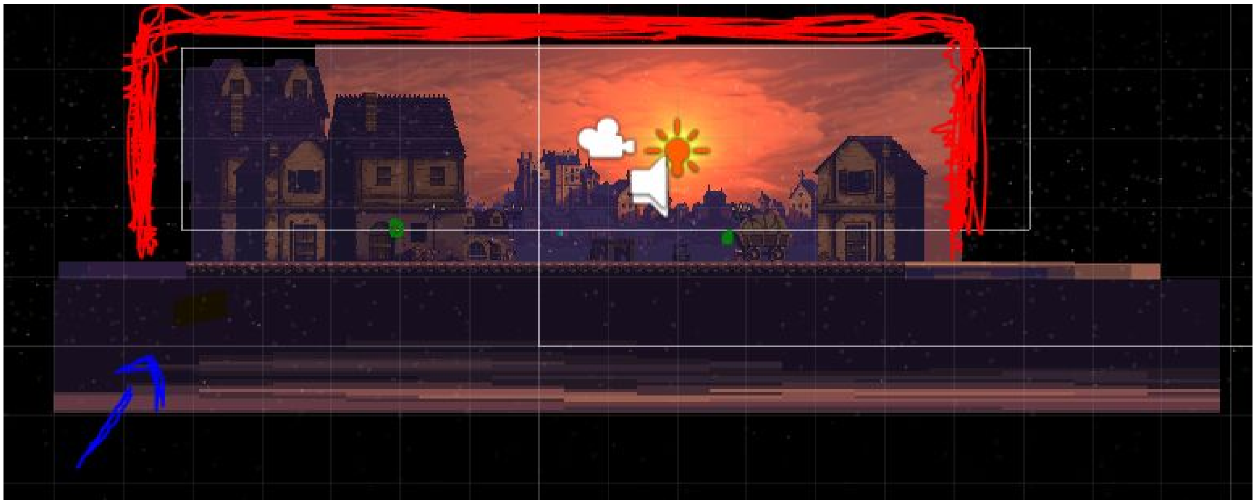
Treant detects collisions by raycasting in front of him. Then takes into account the size of obstacle and tweaks the equation accordingly to make the jump.



```
if (enemy_info.transform.tag == "obstacle")
{
    //velocity of jump in accordance to size of obstacle
    float obstacle_size = enemy_info.transform.GetComponent<SpriteRenderer>().size.y;
    Debug.Log(obstacle_size);

    if (!obstacleCheck) {
        obstacleCheck = true;
        this.GetComponent<Rigidbody2D>().velocity = new Vector2(this.GetComponent<Rigidbody2D>().velocity.x, obstacle_size + 1.5f);
    }
    Debug.Log("jump");
    obstacleCheck = false;
}
```

Map:

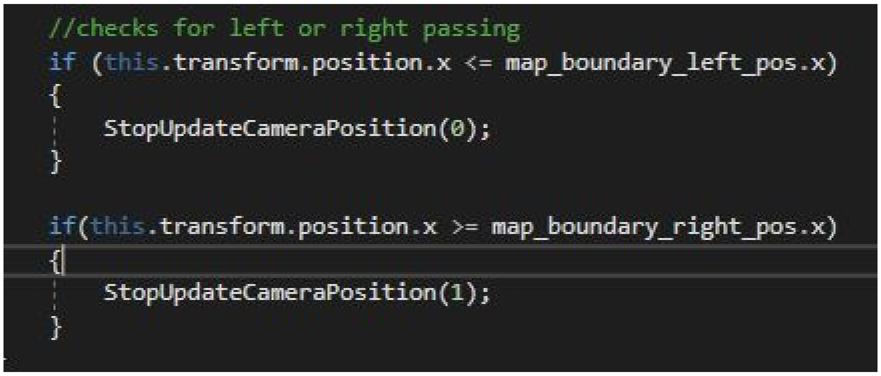
The map consists of sprites layered in {Background, Middleground, Foreground, Sprites, Particle Systems}. The background is controlled by a script for infinite scrolling. The sun is a point light. The sky is illuminated by a directional light, this is used for the explosion effect.



The red is the boundary that inhibits the player from walking off the edges (Boxes with no meshes). The blue arrow is the physical ground.

Camera:

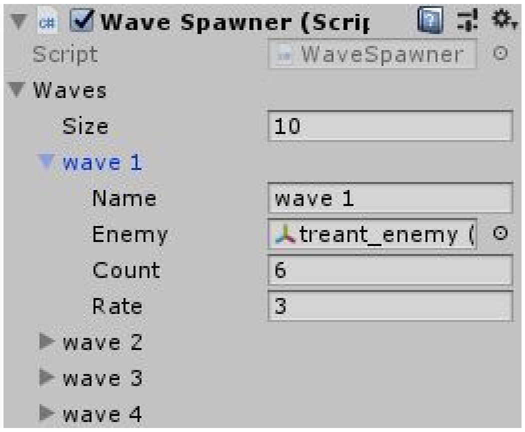
The camera follows the player due to the *CameraFollowScript*. The camera stops following the player when he gets to the edges. The green points show the place of the empty game objects that detect the passing of the player. That gives a nice retro arcade feel.



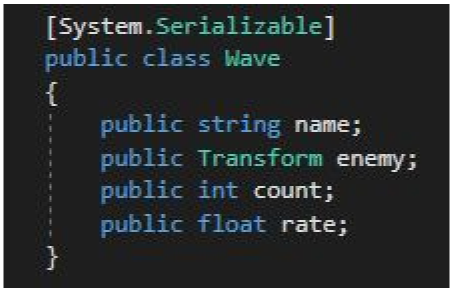
```
//checks for left or right passing
if (this.transform.position.x <= map_boundary_left_pos.x)
{
    StopUpdateCameraPosition(0);
}

if(this.transform.position.x >= map_boundary_right_pos.x)
{
    StopUpdateCameraPosition(1);
}
```

Spawn System:

This is placed on the WaveSpawner empty game object. Spawning is made easy by using a serializable menu.



```
Wave Spawner (Scrip
Script        WaveSpawner
Waves
  Size        10
  wave 1
    Name      wave 1
    Enemy     treant_enemy (
    Count     6
    Rate      3
  wave 2
  wave 3
  wave 4
```



```
[System.Serializable]
public class Wave
{
    public string name;
    public Transform enemy;
    public int count;
    public float rate;
}
```

And controlled by three states, Spawning for when the enemies are spawning, Waiting for when the system is counting to next round and Counting to when the enemies on the scene are being counted.

Game Manager:

The Game Manager script is placed on the GameManager empty game object. It’s role is to handle the UI and maintain Global variables across entire scripts, like player death, sounds and the frenzy effect.