Zachary Briggs
July 2nd, 2018
Assessment Five

1. Assessment Outline

1.1. Goal

Name: Cross-Platform Development

Problem Statement: Create a program that can be played on multiple platforms.

1.2. **Input Information:** PC Controls: W,A,S,D to move, left click to fire, and 1,2,3 to switch weapons. Mobile Controls: Swipe to move, tap to shoot.

1.3. **Output Information:** None.

1.4. **User Interface:** Buttons for changing weapons and timer.

2. Requirements

Support Multiple Platforms

PC and Android Supported. PC build and Android APK provided in submission.

Shooting is handled the similarly between PC and Android. Holding down fire for PC would let the player fire without having to constantly click, since this might conflict with the swipe controls on Android a preprocessing directive is used to have different functionality on different platforms.

```
#if UNITY_ANDROID
       if (Input.GetButtonUp("Fire1") && Timer <= 0)
       {
           weapon.Shoot(transform, centerPos);
           Timer = weapon.FireRate;
       }
#else
    if (Input.GetButton("Fire1") && Timer <= 0)
       {
           weapon.Shoot(transform, centerPos);
           Timer = weapon.FireRate;
       }
#endif
```

Fire1 counts as both a click and a tap so there's no need for any extra code for either platform.

However, movement is handled differently for both platforms.

PC movement is handled in the PlayerBehaviour

```
void Update ()
        {
        var h = Input.GetAxis("Horizontal");
                var v = Input.GetAxis("Vertical");
        Vector2 InputVec = new Vector2(h,v);
        rb2d.AddForce(InputVec * Speed * Time.deltaTime);
         }
```

Since there is no equivalent for the keyboard on a phone screen a SwipeBehaviour is put on the Player. A preprocessing directive is used so that the code for SwipeBehaviour doesn't function on PC.

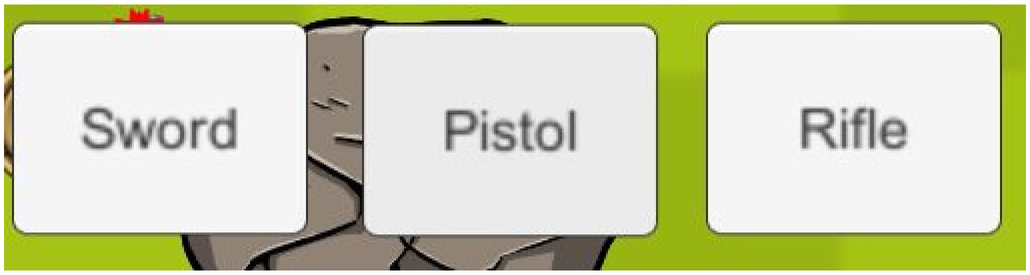
Swipe Controls Behaviour

```
public class SwipeBehaviour : MonoBehaviour
{
   public float Force = 1;
   private Vector2 startPos;
   private Vector2 endPos;
   private Vector2 direction;
   private float touchStart;
   private float touchFinish;
   private float touchInterval;

   // Update is called once per frame
   void Update()
   {
#if UNITY_ANDROID



      if (Input.touchCount > 0 && Input.GetTouch(0).phase ==
TouchPhase.Began)
      {
         touchStart = Time.time;
         startPos = Input.GetTouch(0).position;
      }

      if (Input.touchCount > 0 && Input.GetTouch(0).phase ==
TouchPhase.Ended)
      {
            touchFinish = Time.time;
            touchInterval = touchFinish - touchStart;
            endPos = Input.GetTouch(0).position;
            direction = startPos - endPos;
            Rigidbody2D rb2d = GetComponent<Rigidbody2D>();
            rb2d.AddForce(-direction / touchInterval * Force);
        }
#endif
    }
}
```

1,2,3 is used for changing weapons on PC in the PlayerBehaviour

```
void Update ()
          {
          if (Input.GetButtonDown("Weapon1"))
              {
                  CurrentWeapon.weapon = Inventory.Weapons[0];
              }
              if (Input.GetButtonDown("Weapon2"))
              {
                  CurrentWeapon.weapon = Inventory.Weapons[1];
              }
              if (Input.GetButtonDown("Weapon3"))
              {
                  CurrentWeapon.weapon = Inventory.Weapons[2];
              }
          }
```

On Android the 3 UI buttons are used for changing weapons.



Demonstrate Inheritance

EnemyShootingBehaviour inherits from EnemyBehaviour. Melee scriptable inherits from WeaponScriptable.

WeaponScriptable UML

+Projectile : GameObject
+Force : Int
+FireRate : Float
+Shoot()

MeleeScriptable inherits from WeaponScriptable with the only change being the shoot function.

WeaponScriptable

```
public virtual void Shoot(Transform transform,Transform centerpos)
   {
       Vector2 mousePos =
Camera.main.ScreenToWorldPoint(Input.mousePosition);
       Vector2 currentPos = new Vector2(transform.position.x,
transform.position.y);
       Vector2 direction = mousePos - currentPos;
       Vector2 pos2D = new Vector2();
       pos2D.x = centerpos.position.x;
       pos2D.y = centerpos.position.y;
       direction.Normalize();
       var projectile = Instantiate(Projectile, pos2D + direction,
Quaternion.identity);
       var rb2d = projectile.GetComponent<Rigidbody2D>();
       rb2d.AddForce(direction * Force, ForceMode2D.Impulse);
   }
```

MeleeScriptable is mostly the same except the Rigidbody2D force is not applied.

EnemyShootingBehaviour also inherits from EnemyBehaviour with the main difference being an added Shoot() function along with ewn fields used with the Shoot() function.

EnemyBehaviour UML

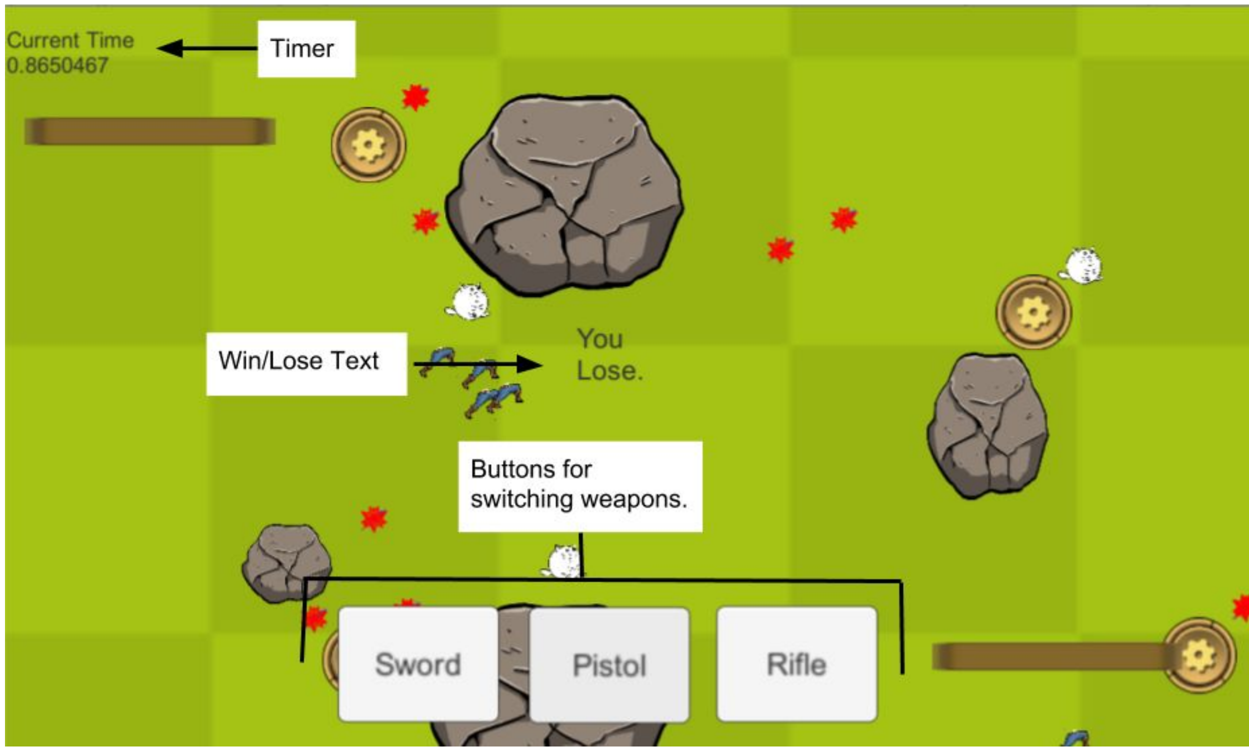
#Player : GameObject
#dirToPlayer : Vector3
+Speed : Float
-Start() : Void

-Update() : Void
-OnCollisionEnter2D(Collision2D) : Void
#Move(float) : Void

EnemyShootBehaviour UML
+ProjectilePrefab : GameObject
+Cooldown : Float
+Force : Float
-Timer : Float
-Start() : Void
-Update() : Void
-Shoot() : Void

Implement a Graphical User Interface

Text is displayed upon winning or losing the game. Buttons are displayed on screen for switching weapons.



Use of Scriptable Objects

Weapon scriptable is used for multiple weapon types.

WeaponScriptable UML

+Projectile : GameObject
+Force : Int

+FireRate : Float
+Shoot()

The pistol has stats like this:
Projectile : PlayerProjectile
Force : 9
FireRate : 0.3

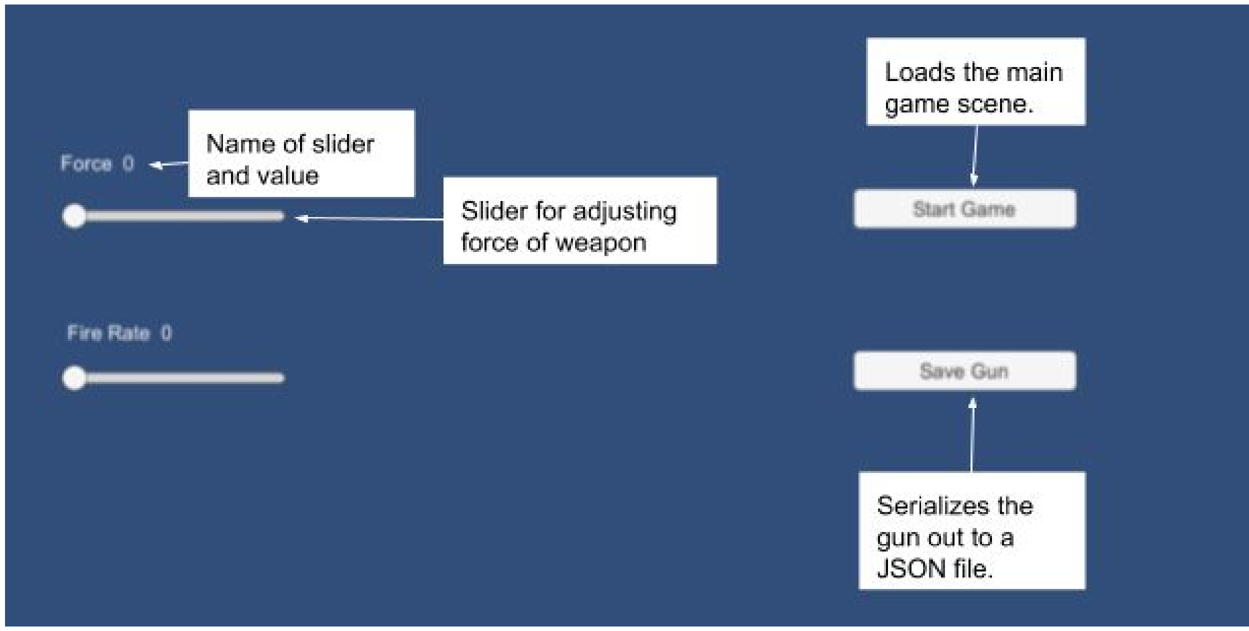
While the rifle has stats like this:
Projectile : RifleBullet
Force : 30
FireRate : 0.9

And the sword has stats like this:
Projectile : SwordProjectile
Force : 1
FireRate : 1
The behaviours assigned to the projectile can also affect how each weapon fires. The sword’s projectile, for example, doesn’t destroy itself on collision like the other two projectiles.

Player’s inventory is also scriptable object just taking in WeaponScriptables.

Use of JSON format to persist data.

Upon starting the game a custom weapon can be created to be used within the game.


Loads the main game scene.
Name of slider and value
Force 0
Slider for adjusting force of weapon
Start Game
Fire Rate 0
Save Gun
Serializes the gun out to a JSON file.

The weapon file is saved out to the StrangeLand->Assets->StreamingAssets folder as a .txt document with the values of the sliders.

If the player has created a custom weapon before that will be load automatically upon starting the game.

WeaponMenuBehaviour.cs

```
void Start ()
   {
      filePath = Path.Combine(Application.streamingAssetsPath, FileName);
      if (File.Exists(filePath))
      {
         string data = File.ReadAllText(filePath);
         ClassWeapon = JsonUtility.FromJson<WeaponClass>(data);
      }
      else
      {
         string data = JsonUtility.ToJson(ClassWeapon);
         File.WriteAllText(filePath, data);
      }
```

```
        ForceText = ForceSlider.GetComponentInChildren<Text>();
        FireRateText = FireRateSlider.GetComponentInChildren<Text>();
    }
```

Unity will throw errors when a Scriptable Object JSON is load back in so the file saved isn't a Scriptable Object in JSON format, instead it is a C# class that represents the Scriptable Object.

```
public class WeaponClass
{
    public int Force;
    public float FireRate;

}
```

When the WeaponClass JSON is loaded back in it's values are plugged into a WeaponScriptable that is passed in. The WeaponScriptable that is passed in this game is the player's 2nd weapon.

GameManagerBehaviour.cs

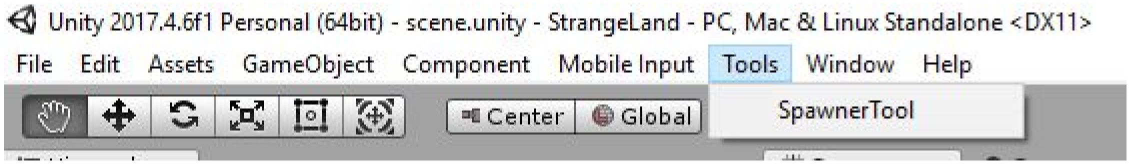
```
public class GameManagerBehaviour : MonoBehaviour
{
    public WeaponScriptable CustomWeapon;
    private float TimePassed = 0;
    private GameObject Player;
    public Text CurrentText;
    public GameObject winText;
    public GameObject loseText;
    private string path =
System.IO.Path.Combine(Environment.CurrentDirectory, "time.txt");

    void Start()
    {
        WeaponClass ClassWeapon = new WeaponClass();
        ClassWeapon.Force = 9;
        ClassWeapon.FireRate = 0.3f;
        string filePath = Path.Combine(Application.streamingAssetsPath,
"Weapon.txt");
        if (File.Exists(filePath))
        {
            string data = File.ReadAllText(filePath);
            ClassWeapon = JsonUtility.FromJson<WeaponClass>(data);
        }
        else
        {
            string data = JsonUtility.ToJson(ClassWeapon);
            File.WriteAllText(filePath, data);
        }

        CustomWeapon.Force = ClassWeapon.Force;
        CustomWeapon.FireRate = ClassWeapon.FireRate;
    }
}
```

User of Editor Tool that builds objects.

Prefab spawner for placing and moving enemies or other prefabs.

How to use

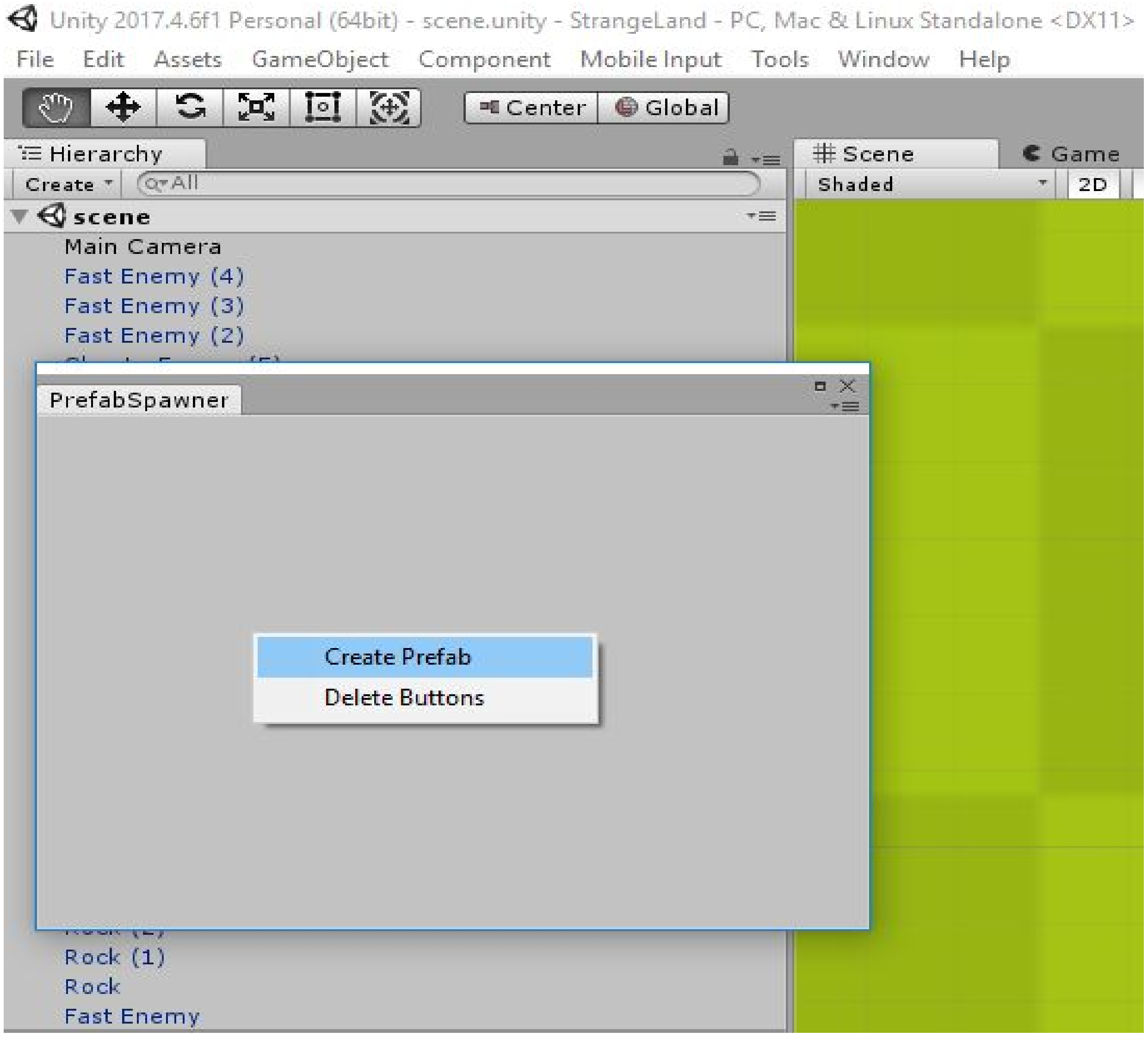
1. Click the Tools button at the top of the Unity window.
2. Click Spawner Tool. This will open the window for the prefab spawner.



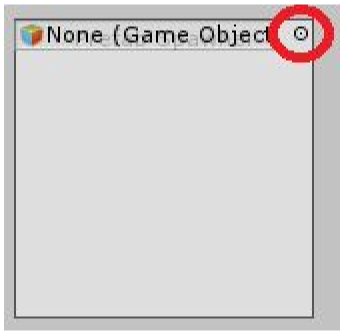
Unity 2017.4.6f1 Personal (64bit) - scene.unity - StrangeLand - PC, Mac & Linux Standalone <DX11>
File
Edit
Assets
GameObject
Component
Mobile Input
Tools
Window
Help
Center
Global
SpawnerTool

3. In the window that opened, right click to open the context menu.
4. Click the “Create Prefab” button. A new window will open for selecting the prefab.

If you want to clear the window of all the prefabs press the “Delete Buttons” button.

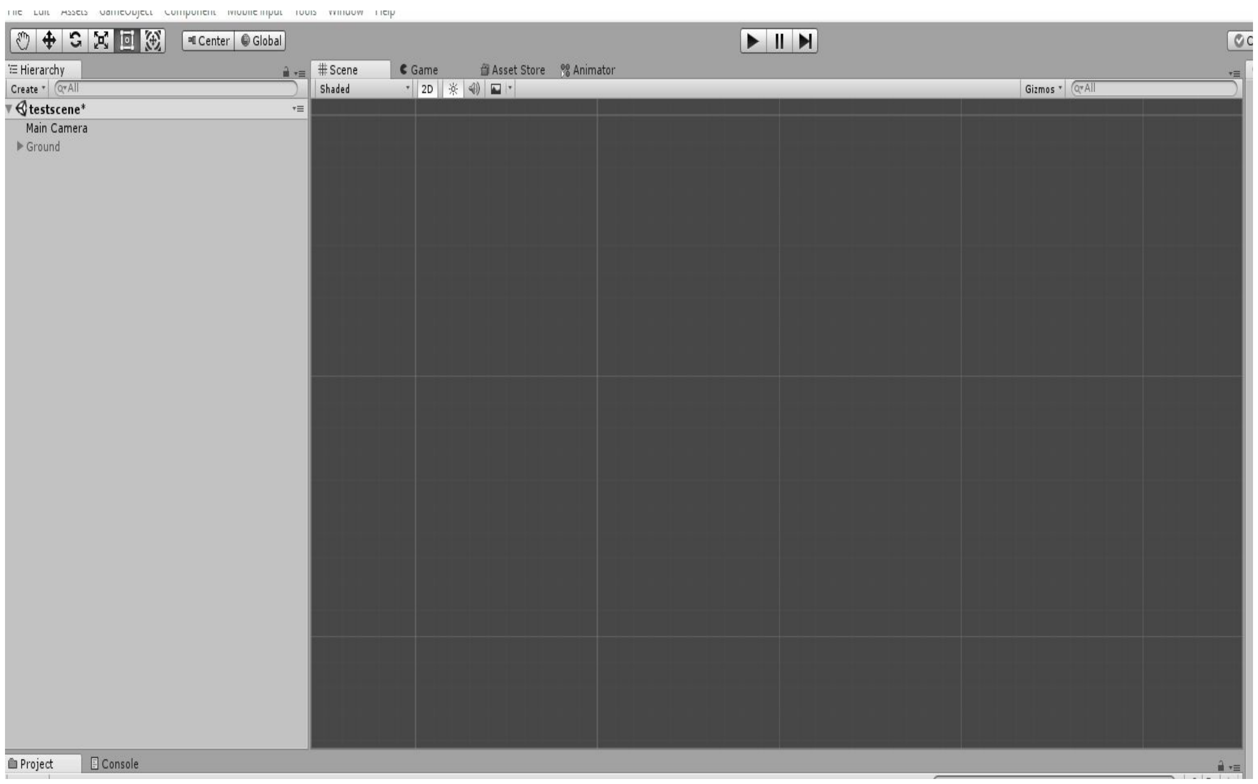


5. Click the circle icon at the top of the new window.



6. Another window will open with a list of prefabs.
7. Double click the prefab you wish to spawn.

8. Once the prefab is spawned the window can be used to move the prefab. The prefab will be at the position noted in the window.



3. Read Me

Repo: https://github.com/ZacharyBriggs/StrangeLand

PC Controls: W,A,S,D to move. Left click shoot. 1 changes weapon to sword, 2 changes weapon to pistol, 3 changes weapon to sniper rifle.

Android Controls: Swipe to move, Tap to shoot, press 1,2,3 buttons to switch weapons.