

# Work Log:

# Alexander Freyr Þorgeirsson

Project Team: 5

Date of check-in: 07/02/24

# Design for Enemy Spawning-System

Date of completion:  
02/03/2024

Approximate time taken: 2h

Brief description of task (10 – 30 words):

Designed (not implemented yet) a spawning system that ensures enemies are equally spaced out in a very elegant and flexible manner.

This whole thing is a problem if enemies move at different speeds

Unless I write an incredibly complicated function that calculates: if I would spawn an enemy with a certain speed, what distance would it travel before reaching the prior enemy. If the distance is over the height of the screen then I can spawn it, knowing they would collide outside the bottom of the screen.

Each spawn point stores the previously spawned enemy and the negative time since it was able to spawn again (basically telling you where its top part is ATM)

This system would ensure that the spawning enemies would be fixed to a grid which ensures they will be spaced out to not awkwardly intersect with each other which is both more visually appealing and removes cases where the player gets an unfair, inescapable spawning formation

$\text{float gridWidth} \Rightarrow \frac{\text{Camera.orthographicScale}}{\text{gridCount}}$   
 $\text{int gridCount}$

each spawn point has a timer which is set to the time-out duration when spawned over, which makes it not possible to be spawned over again during that time

Will be the top reference of the spawn prefab

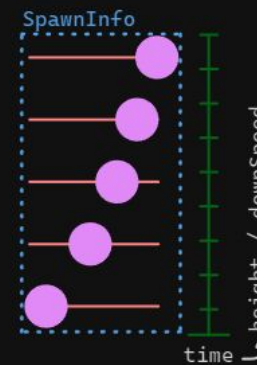
SpawnInfo

Public float TimeOutDuration  
public float Width, Height  
public float DownSpeed

TimeOutDuration will be calculated according to the height of the spawn and the down speed

Width tells you how many neighboring spawn points will receive time out

$\text{neighboringCount} = \text{floor}(\text{Width} / \text{gridWidth})$



# “Ideas and unanswered design questions” documentation

Date of completion: February 1




Approximate time taken: 40min








Brief description of task (10 – 30 words):

Created a section in our design document for design ideas and unanswered questions

Link to external document (if applicable):

[Ideas and unanswered design questions](#)

**New Ideas and Unanswered design question** // Decided suggestions are either  Approved and stored or removed in the case of  declined or  implemented

-  Open suggestion  
Enemy behind: When it's within a certain closeness an alert comes up that within x seconds (~2s) it will bite you from behind and you have to dash forward to dodge it but also keep in mind what's ahead of you.
-  Open suggestion  
The large monster can appear after a few seconds, not there from the very beginning.
-  Open suggestion  
Boost forward to go advance through the game quicker and avoid the monster from behind
-  Open suggestion  
Harpoon (has to be reeled in with down arrow) or missile (blast radius damages you if you hit an enemy that's too close) as powerful attack
-  Open suggestion  
Square game world to more simulate the vastness of the deep sea.
-  Open suggestion  
For tutorial: different enemies are introduced in the first section and throughout the first section your shooting is disabled. It is then unlocked through a funny comic that shows the player discovering the weapon.
-  Open suggestion  
For tutorial: Maybe there are two submarines on this mission, then one of them hits a fish and explodes (for shock and comedic value), showing that fish kill.

# Website for playtesters

Date of completion: 06/02/24

Approximate time taken: 3h

Brief description of task (10 – 30 words):

Created a website that hosts all game variations of our game so we can let people playtest more than one version, due to the limitation of Itch.io

Link to external document (if applicable):

<https://blog.alexanderfreyr.com/Splish-Splash-Mods/>

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Splish Splash Submarine</title>
    <link rel="stylesheet" href="./TemplateData/style.css" />
  </head>
  <body>
    <h1>Game Modes</h1>
    <ul>
```

## Game Modes : skill issue basic

```
.a {
  font-weight: bold;
  font-size: 2rem;
  color: #70a1ff;
}
li {
  font-weight: bold;
  font-size: 2rem;
}
</style>
</html>
```



# Enemy Movement Pattern

Date of completion: 2024-01-29

Approximate time taken: 3h

Brief description of task (10 – 30 words):

Created a comprehensive enemy movement script using only sine waves and phase offsets. I also created a live preview of the pattern through red lines.

Unity Message | 0 references

`void OnDrawGizmos()`

`{`

`Gizmos.color = Color.red;`

`int lineCount = (int)(lineLength * m`

`Vector3 lastPoint = Evaluate(0) + tra`

`for (int i = 1; i < lineCount; i++)`

`{`

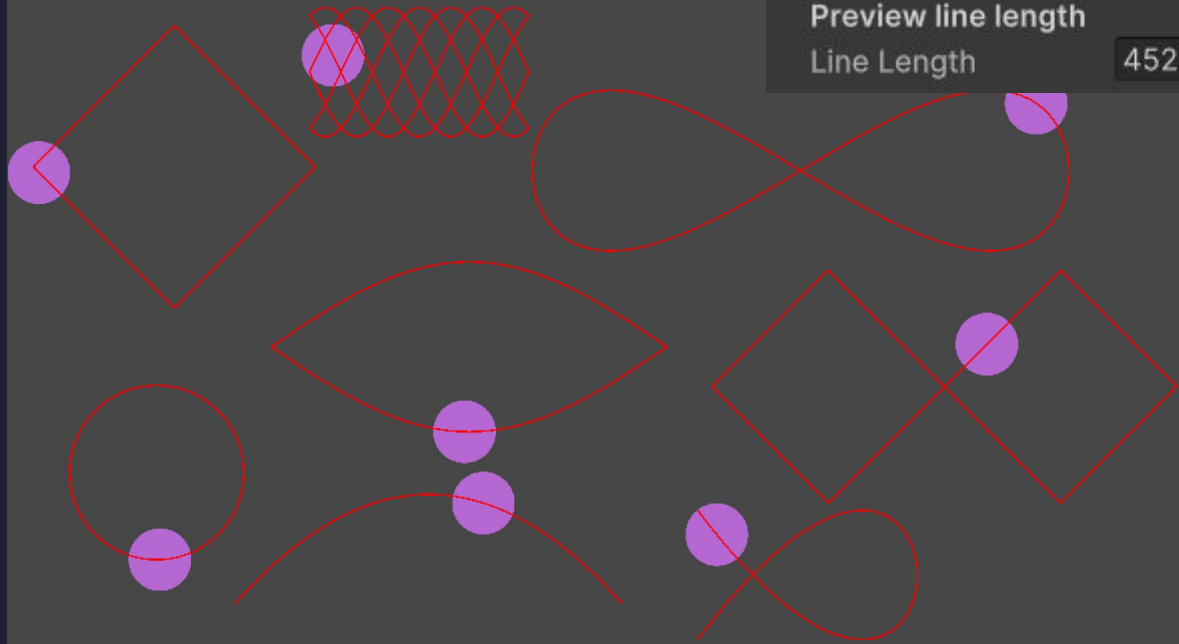
`Vector3 point = Evaluate(i / (lin`

`Gizmos.DrawLine(lastPoint, point);`

`lastPoint = point;`

`}`

`}`



Script

MovePattern

Magnitude

5

Frequency

2

Magnitude Ratio

0.66666

Frequency Ratio

0.33333

Timing Offset

0

Phase Offset

0.25

Linear Interpolation

OFF: circle, ON: linear

Linear X

☒

Linear Interpolation

OFF: circle, ON: linear

Linear Y

☒

Gizmos

Preview line length

Line Length

452.8

# Global Game Settings

Date of completion: 03/02/24

Approximate time taken: 13h

**Brief description of task (10 – 30 words):**

Designing and implementing a hierarchy of scriptable objects that can store settings, modes and variants for easy access to ALL variables and to both speed up and force structure to iterative design.

```
[CanEditMultipleObjects]
[CustomEditor(typeof(ScriptableObject), true)]
// Unity Script | 0 references
public class ScriptableObjectEditor : Editor
{
    // 0 references
    public override void OnInspectorGUI()
    {
        if (serializedObject == null)
            return;

        DrawPropertiesExcluding(serializedObject, "m_Script");

        if (GUI.changed)
            EditorUtility.SetDirty(target);
        serializedObject.ApplyModifiedProperties();
        serializedObject.Update();

        EditorGUILayout.Separator();
    }

    [MenuItem("Game Settings/Open Global Settings")]
    // 0 references
    public static void OpenGlobalSettings()
    {
        EditorUtility.OpenPropertyEditor(GlobalSe
    }
}

[CustomPropertyDrawer(typeof(ScriptableObject), t
// 0 references
public class ScriptableObjectPropertyDrawer : Pro
{
    // 0 references
    public override void OnGUI(Rect position, Ser
    {
        EditorGUI.PropertyField(position, property

        if (property.objectReferenceValue != null)
            Editor.CreateEditor(property.objectRe

        property.serializedObject.ApplyModifiedPr
        property.serializedObject.Update();
    }
}

// Unity Script | 20 references
public class GlobalSettings : ScriptableObject
{
    // 4 references
    public static GlobalSettings Get => (_instance != null)
        ? _instance
        : (_instance = LoadGameSettings());
    static GlobalSettings _instance;

    const string Path = "Settings/GlobalSettings";
    const string FullPath = "Assets/Resources/" + Path + ".asset"
    // 0 references
    public static ScriptableObject Load(string name) => Resource
    // 10 references
    public static GameSettings Current => Get._current;
    [Tooltip("The currently selected game mode setting. This is
    public GameSettings _current;

    #if UNITY_EDITOR
    // 1 reference
    static GlobalSettings LoadGameSettings()
    {
        GlobalSettings settings = Resources.Load<GlobalSettings>

        // if settings is null, create a new one
        if (settings == null)
        {
            settings = CreateInstance<GlobalSettings>();
            UnityEditor.AssetDatabase.CreateAsset(settings, Path
            UnityEditor.AssetDatabase.SaveAssets();
            Debug.LogError($"{No (nameof(GlobalSettings)) found a
        }

        return settings;
    }
    #else
    static GlobalSettings LoadGameSettings() => Resources.Load<G
    #endif
}

// Unity Script | 5 references
public class SettingsBase<T> : ScriptableObject where T : Settin
{
    public Action<T> onValidate;

    public const string Path = "Setting Objects/";

    // Unity Message | 0 references
    void OnValidate()
    {
        onValidate?.Invoke((T)this);
    }
}
```

This will enable the designer to create drastically different gameplay from mixing and matching different variables to both quickly and easily test out gameplay variants and to save to a single settings object for later use

