



AI FOR GAMES COS30002

Custom Project Plan

The following document has been prepared as the Custom Task, this document contains resources that can be used to implement the features added to this 2d game designed in unity with their reference for learning.

Hasaan Akhtar

102400615

Overview

The following document would contain some implementation and design for a 2d game but keep in mind the following prefab for the original game design has been taken from the following GitHub repository and the design credit for the environment belongs to the author and not me but implementation of the key features such as Animation added to characters, FSM (finite state machine diagram), Character movement restricted to screen boundary, XP gauge, Path finding algorithm and other features such as Auto Spawn game objects, Delete Game Objects has been added and the reference have been marked for each as to how I learned implementing them plus the original Environment design I got from the Brackeys Game tutorial with unity.

Original GitHub repository link:

GitHub. (2020). Brackeys/2D-Movement. [online] Available at:

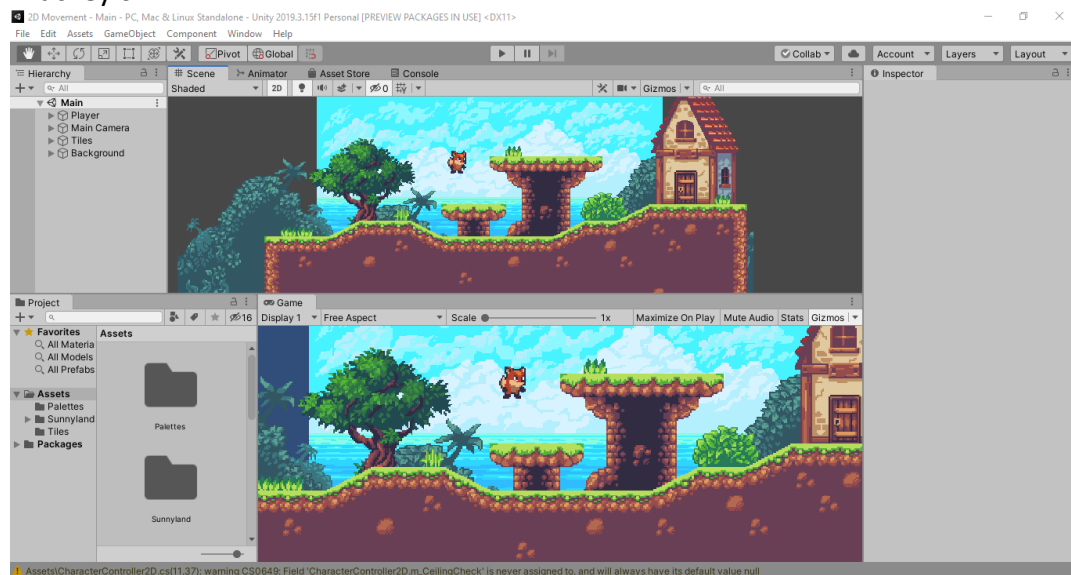
<https://github.com/Brackeys/2D-Movement.git> [Accessed 10 Jun. 2020].

My repository link:

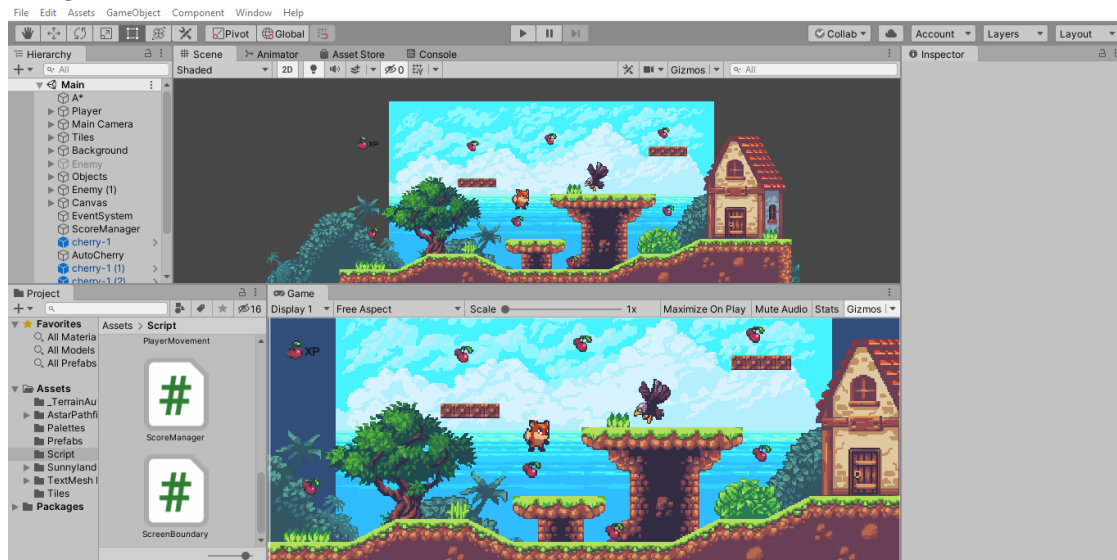
<https://github.com/HasaanAkhtar/EndlessChase.git>

My Implementation of Environment for the game as well as the original one I used from GitHub (Brackey's work is all used for educational and learning purpose and no official copyright has been claimed) can be seen below.

Brackey's:



Mine:



The changes to the environment such as objects added (cherry for the player to collect), two moving and floating rectangular blocks plus the eagle/enemy choosing the best/shortest path to player using the A* path finding algorithm as well as all the scripts can be seen all in this document.

The above game what I originally intend to call it was 'Endless Chase' but now I want to edit it's environment to look more like 'Sonic' from the old days though since this custom project was suppose to be more of a plan than an actual implementation I didn't implmenet much things in it but what I did add was some of the stuff that I learned quite well from the unit it self 'AI for games' I intended to do more implmentation in my D project which I'm still not sure if I would go for or not.

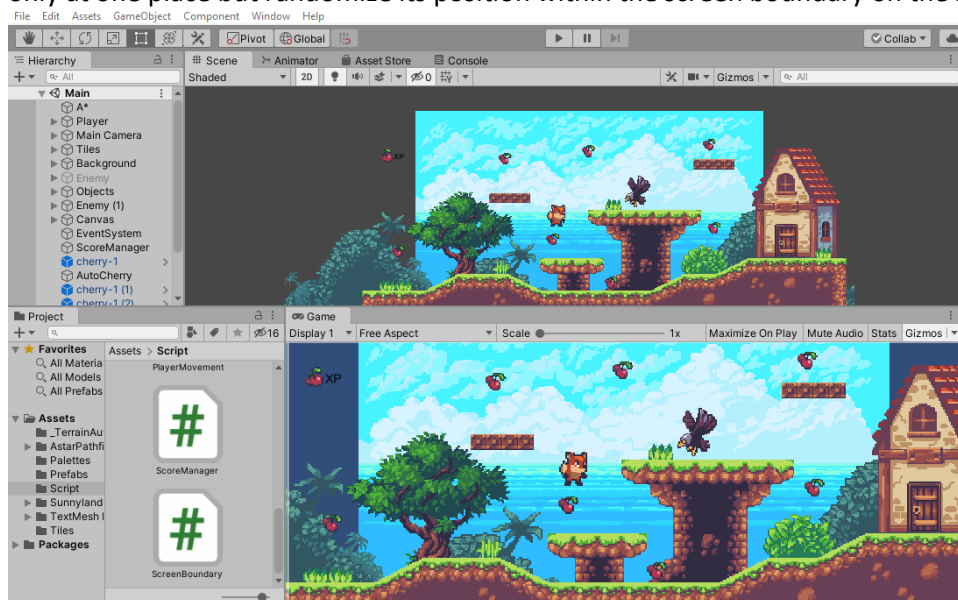
From its appearance it can be made apparent that the game is obviously a 2d game which means that it utilizes only two vector dimension the X-axis and the Y-axis for the character movements mostly since there is no 3d object in the game. A 2d game was way more easier to work with at this level since I'm a total beginner with unity and I had to spent entire 2 weeks to get used to unity, I really didn't wanted to work with 3d right from the start.

Game Objectives and Supportives:

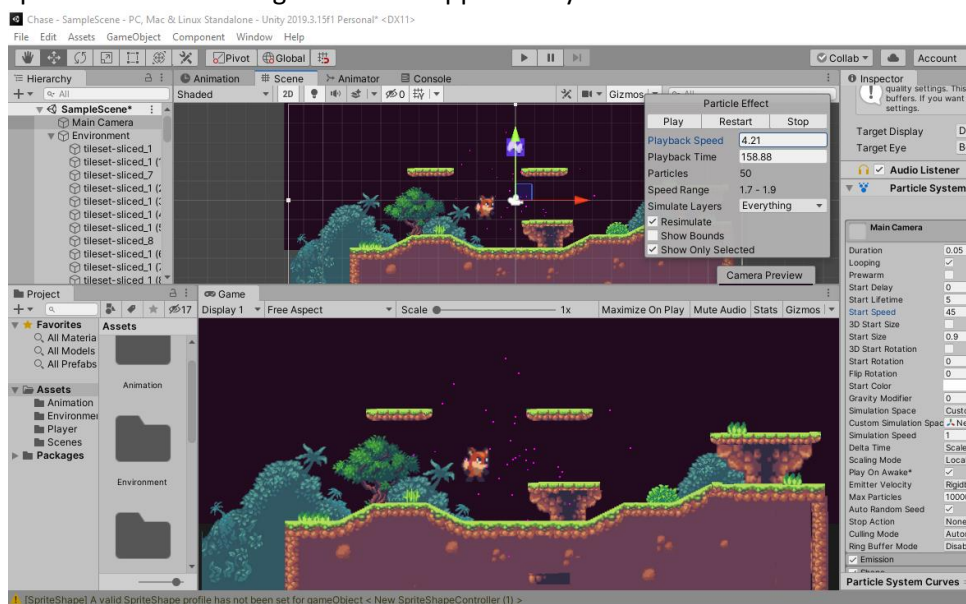
The character is lost in a deserted place and tries hard to survive by munching on the resources available to him but with the obstacle i.e. the enemy eagle which tries to guard the survival resources from the character who hunts for them, the enemy(eagle, possum, frog) follows the AI patrol and Chase game concept from the Task 08 - Lab - Steering #1 - Seek, Arrive, Flee, Task 09 - Lab - Steering #2 - Wander and Paths, Task 10 - Spike - Tactical Steering (Hiding) and lastly Task 15 - Spike – Soldier On Patrol. And as the name of each task gives away the reason logic behind them I won't really go into details but the reason why I chose this particular tasks were because of the story it plays out and the necessary features for the game.

Environment with different stages:

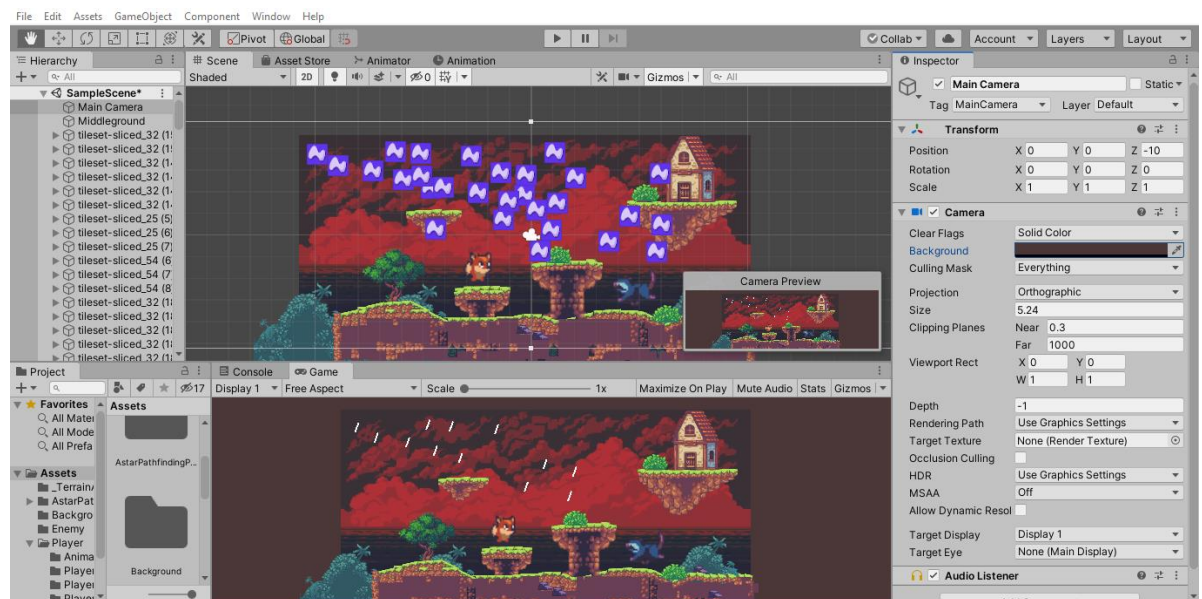
1. Level 1, Initial day Environment where player finds himself on deserted land and encounter the eagle enemy while searching for food to survive, the gauge at the top right indicates the number of cherries collected, the way this works is that I used On collider trigger event to detect the collision with the player with tags and then delete game objects from the scene but doing so also posed a problem and that was if each object was being deleted from the scene and the Score Manager was keeping a record of the items deleted, there would definitely come a time when no object would left on the scene to avoid that state I used the Auto Spawn script to add the objects within the screen boundary and with create timer plus randomise position (did in the tasks on doubtfire) to help so that the Cherry doesn't pop only at one place but randomize its position within the screen boundary on the scene.



2. Level 2, Night Environment with level 2 hyper dimension (a weird concept but the player intends to do image training in his head to counter eagle as the enemy). Used Particle Effect for the hyper dimension look and rigid2d body for the floating rocks in the dimensional space and character's ground was supported by the box collider.



- Level 3 Acid rain which also decreases player health and is caused by opossum/enemy character who tries to stop you from seeking shelter from the storm inside the floating castle (more of a house the castle was expensive to buy at this stage). As it can be clearly seen the acid rain is directed towards fox player. For this implementation I used collider and put an ontrigger event that whenever it collides with the player it would decrease player hp level.



Character Actions and roles:

Player:

- Move about horizontally.
- Jump, double Jump.
- Crouch to hide under objects and pass through narrow areas.
- Attack enemies but flee if enemy is stronger (used the logic from Task15)
- Collect Cherries to survive and leave the deserted Island.

















Enemy:



- If bird then fly and attack player character, flying enabled with rigid 2d body anti-gravity mode, if frog then hop towards the player, and if opossum then chase on ground enemy.
- Bird can fly, frog can hop and do double hop in order to hop at higher position, and opossum can jump.
- None can crouch.
- Attacking player based on their natural hunting skills, for instance with birds it's the claws and its mouth peak, with frog it's the tongue and with the opossum it's the claws and dash attacks.
- The reason they hinder the fox from reaching to next level is quite dramatic, it's because the eagle can have a god's eye view from the sky and knows that nothing exists beyond the Island grounds except ocean, with frog it's because it wants to discover an exotic new taste of devouring the fox but with opossum is that it wants to stop the fox from going inside the floating castle/house because a beast awaits there stronger than anyone on the Island.

How it all works:

- Attacks: they work via the Collider events trigger and collides with Object Tag under either player or enemy.
- Jump: Anti-gravity for the part when player jump's and the distance is calculated based on the number of times the key is pressed within the time frame.
- Horizontal movement is all based on the rigid2d body which is basically physics 2d and then the vector position is calculated based on the arrow keys and as for the speed the `time.deltaTime` is used which basically indicates the real time and it's same for all CPUS no matter the processing power so that with fast CPU the player doesn't fly off or neither it lags with the slower ones.
- With the Path following I used A* AI path finding and Seeker script so that the shortest and most effective route is always chosen to chase after the player plus the distance where it should stop before submissively running into the player.

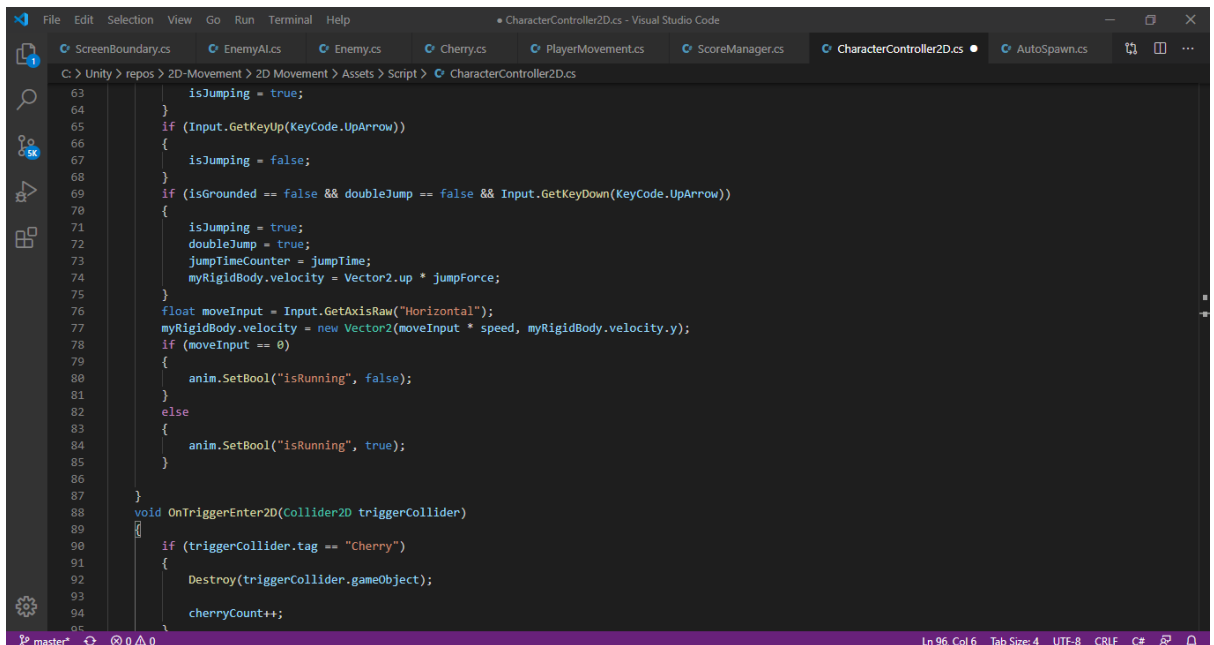
Scripts used and their purpose in this game (Keep in mind this is all actually implemented)

Name	Date modified	Type	Size
 AutoSpawn	9/06/2020 9:50 PM	C# Source File	2 KB
 AutoSpawn.cs.meta	9/06/2020 6:57 PM	META File	1 KB
 CharacterController2D	9/06/2020 7:12 PM	C# Source File	7 KB
 CharacterController2D.cs.meta	8/06/2020 8:41 PM	META File	1 KB
 Cherry	9/06/2020 6:39 PM	C# Source File	1 KB
 Cherry.cs.meta	9/06/2020 5:18 PM	META File	1 KB
 Enemy	9/06/2020 2:11 AM	C# Source File	1 KB
 Enemy.cs.meta	9/06/2020 1:25 AM	META File	1 KB
 EnemyAI	9/06/2020 5:00 PM	C# Source File	2 KB
 EnemyAI.cs.meta	9/06/2020 2:10 AM	META File	1 KB
 PlayerMovement	9/06/2020 4:47 PM	C# Source File	1 KB
 PlayerMovement.cs.meta	8/06/2020 8:41 PM	META File	1 KB
 ScoreManager	9/06/2020 5:58 PM	C# Source File	1 KB
 ScoreManager.cs.meta	9/06/2020 5:13 PM	META File	1 KB
 ScreenBoundary	9/06/2020 7:11 PM	C# Source File	2 KB
 ScreenBoundary.cs.meta	9/06/2020 6:10 PM	META File	1 KB

Name	Date modified	Type	Size
 cherry-1.prefab	9/06/2020 9:35 PM	PREFAB File	5 KB
 cherry-1.prefab.meta	9/06/2020 9:33 PM	META File	1 KB

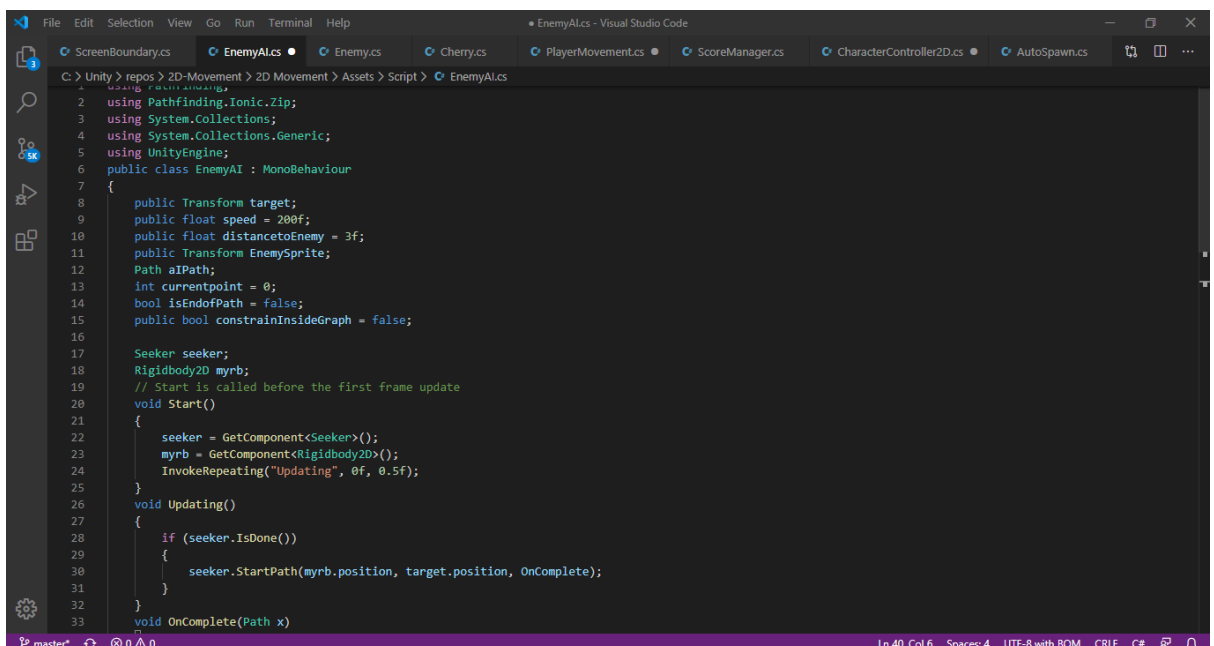
Scripts attached to Player:

1. Character2DController: Manages the state such as Crouching, Jumping, on ground, as well as if the player came in contact with the Cherry object if yes delete the cherry object and increase the Cherry acquired count, also it manages the animation by putting animation with the state in which the object is while also flipping the object sprite based on the vector position. Plus the problem I faced with the character was that if you try to go back to idle state from crouch state while under a narrow hiding object it would create a problem so make sure to include the disable Crouch if collided with the ceiling of the object while transitioning states.



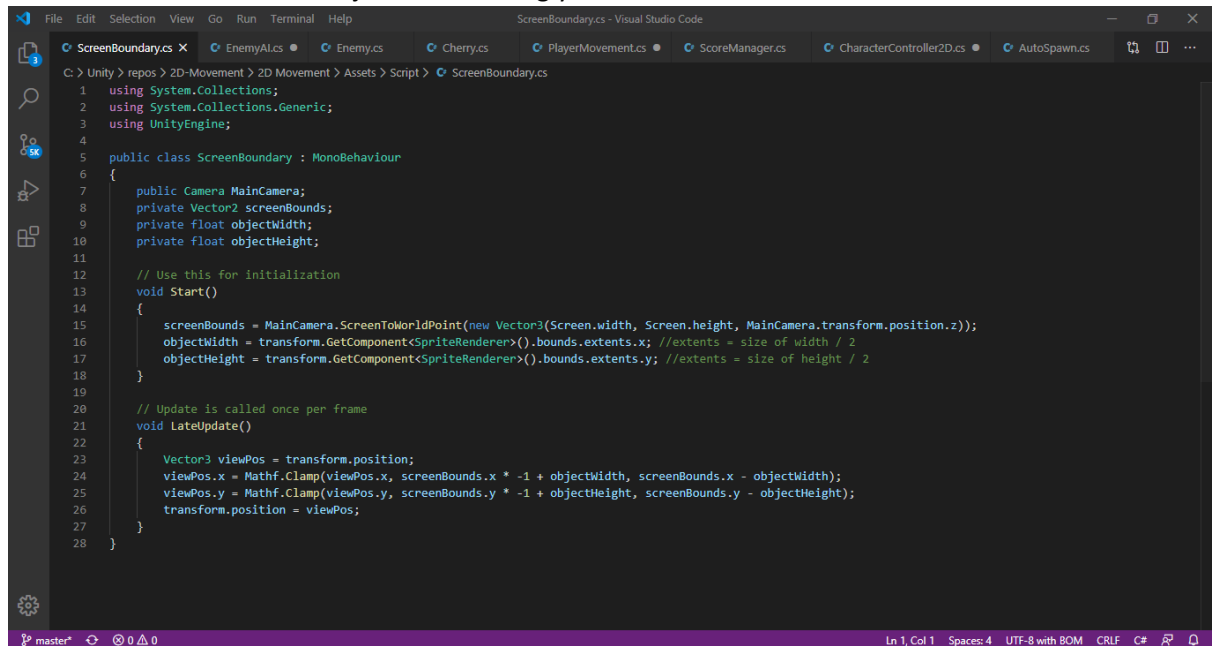
```
File Edit Selection View Go Run Terminal Help
* CharacterController2D.cs - Visual Studio Code
ScreenBoundary.cs EnemyAI.cs Enemy.cs Cherry.cs PlayerMovement.cs ScoreManager.cs CharacterController2D.cs AutoSpawn.cs
C:\> Unity > repos > 2D-Movement > 2D Movement > Assets > Script > CharacterController2D.cs
63     isJumping = true;
64     }
65     if (Input.GetKeyUp(KeyCode.UpArrow))
66     {
67         isJumping = false;
68     }
69     if (isGrounded == false && doubleJump == false && Input.GetKeyDown(KeyCode.UpArrow))
70     {
71         isJumping = true;
72         doubleJump = true;
73         jumpTimeCounter = jumpTime;
74         myRigidBody.velocity = Vector2.up * jumpForce;
75     }
76     float moveInput = Input.GetAxisRaw("Horizontal");
77     myRigidBody.velocity = new Vector2(moveInput * speed, myRigidBody.velocity.y);
78     if (moveInput == 0)
79     {
80         anim.SetBool("isRunning", false);
81     }
82     else
83     {
84         anim.SetBool("isRunning", true);
85     }
86 }
87 void OnTriggerEnter2D(Collider2D triggerCollider)
88 {
89     if (triggerCollider.tag == "Cherry")
90     {
91         Destroy(triggerCollider.gameObject);
92         cherryCount++;
93     }
94 }
```

2. PlayerMovement: Normal player movements such as moving left or right and whether the jump has been triggered if yes use the Controller to pass the Boolean trigger so that it can jump or crouch, since this is a 2d game it's better to go with fixed update instead of instant updates so that the frame rate is balanced and does not hinders the performance of the game.
3. EnemyAI: Uses the AI path to follow the player and seeks the enemy and if detected carry out attack action different for every enemy.



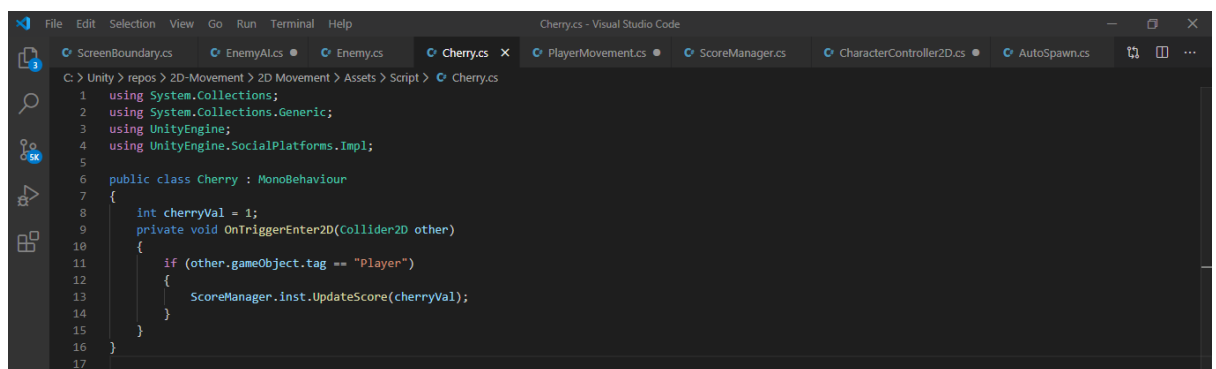
```
File Edit Selection View Go Run Terminal Help
* EnemyAI.cs - Visual Studio Code
ScreenBoundary.cs EnemyAI.cs Enemy.cs Cherry.cs PlayerMovement.cs ScoreManager.cs CharacterController2D.cs AutoSpawn.cs
C:\> Unity > repos > 2D-Movement > 2D Movement > Assets > Script > EnemyAI.cs
1 using Pathfinding.Ionic.Zip;
2 using System.Collections;
3 using System.Collections.Generic;
4 using UnityEngine;
5 using UnityEngine;
6 public class EnemyAI : MonoBehaviour
7 {
8     public Transform target;
9     public float speed = 200f;
10    public float distanceToEnemy = 3f;
11    public Transform enemySprite;
12    Path aIPath;
13    int currentpoint = 0;
14    bool isEndofPath = false;
15    public bool constrainInsideGraph = false;
16
17    Seeker seeker;
18    Rigidbody2D myrb;
19    // Start is called before the first frame update
20    void Start()
21    {
22        seeker = GetComponent<Seeker>();
23        myrb = GetComponent<Rigidbody2D>();
24        InvokeRepeating("Updating", 0f, 0.5f);
25    }
26    void Updating()
27    {
28        if (seeker.IsDone())
29        {
30            seeker.StartPath(myrb.position, target.position, OnComplete);
31        }
32    }
33    void OnComplete(Path x)
```

4. ScreenBoundary: used to make sure that the objects, characters either be player or enemy stay within the screen bounds plus it uses the camera to divide the original screen so no matter which device you play this game on whether it be a cell phone, PC, or even a tablet each screen size would be adjusted accordingly.



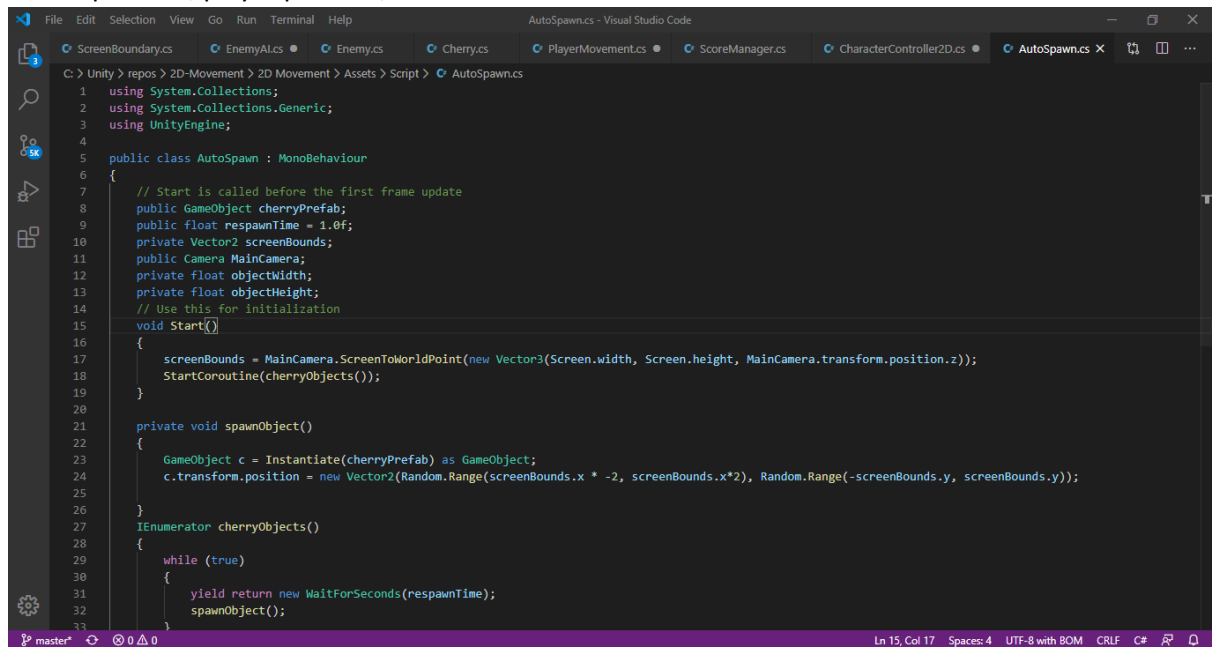
```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ScreenBoundary : MonoBehaviour
6 {
7     public Camera MainCamera;
8     private Vector2 screenBounds;
9     private float objectWidth;
10    private float objectHeight;
11
12    // Use this for initialization
13    void Start()
14    {
15        screenBounds = MainCamera.ScreenToWorldPoint(new Vector3(Screen.width, Screen.height, MainCamera.transform.position.z));
16        objectWidth = transform.GetComponent().bounds.extents.x; //extents = size of width / 2
17        objectHeight = transform.GetComponent().bounds.extents.y; //extents = size of height / 2
18    }
19
20    // Update is called once per frame
21    void LateUpdate()
22    {
23        Vector3 viewPos = transform.position;
24        viewPos.x = Mathf.Clamp(viewPos.x, screenBounds.x * -1 + objectWidth, screenBounds.x + objectWidth);
25        viewPos.y = Mathf.Clamp(viewPos.y, screenBounds.y * -1 + objectHeight, screenBounds.y - objectHeight);
26        transform.position = viewPos;
27    }
28 }
```

5. Cherry: to call the score manager to update the score each time the Player collides with a cherry, the cherry scripts checks if the character that collided with cherry is indeed the player and not the enemy if yes then instantiate the Score Manager script.



```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SocialPlatforms.Impl;
5
6 public class Cherry : MonoBehaviour
7 {
8     int cherryVal = 1;
9     private void OnTriggerEnter2D(Collider2D other)
10    {
11        if (other.gameObject.tag == "Player")
12        {
13            ScoreManager.inst.UpdateScore(cherryVal);
14        }
15    }
16 }
17
```


6. AutoSpawn: To randomise the auto spawn of cherry on to the scene while calculating the collect position, player position, screen boundaries and obstacles on the scene.



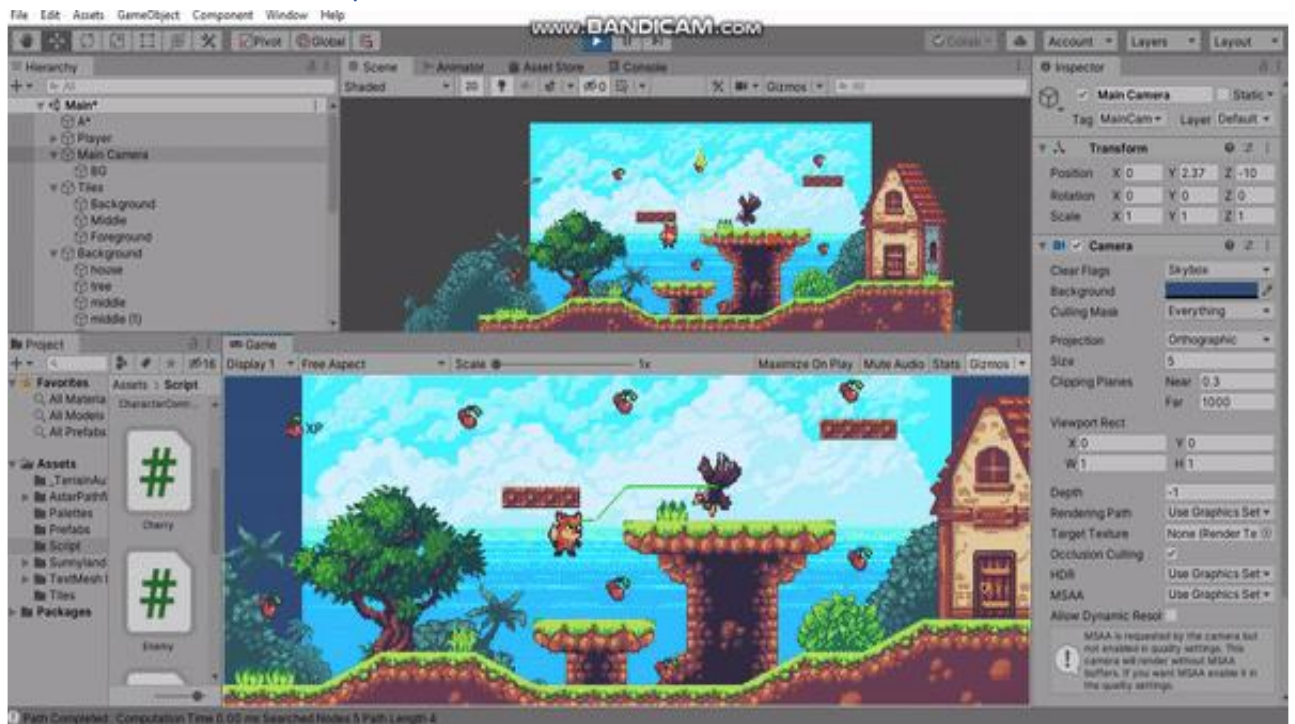
The screenshot shows the Visual Studio Code editor with the 'AutoSpawn.cs' file open. The file is located at 'C:\> Unity > repos > 2D-Movement > 2D Movement > Assets > Script > AutoSpawn.cs'. The code is as follows:

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class AutoSpawn : MonoBehaviour
6 {
7     // Start is called before the first frame update
8     public GameObject cherryPrefab;
9     public float respawnTime = 1.0f;
10    private Vector2 screenBounds;
11    public Camera MainCamera;
12    private float objectWidth;
13    private float objectHeight;
14    // Use this for initialization
15    void Start()
16    {
17        screenBounds = MainCamera.ScreenToWorldPoint(new Vector3(Screen.width, Screen.height, MainCamera.transform.position.z));
18        StartCoroutine(cherryObjects());
19    }
20
21    private void spawnObject()
22    {
23        GameObject c = Instantiate(cherryPrefab) as GameObject;
24        c.transform.position = new Vector2(Random.Range(screenBounds.x * -2, screenBounds.x*2), Random.Range(-screenBounds.y, screenBounds.y));
25    }
26
27    IEnumerator cherryObjects()
28    {
29        while (true)
30        {
31            yield return new WaitForSeconds(respawnTime);
32            spawnObject();
33        }
34    }
35 }
```

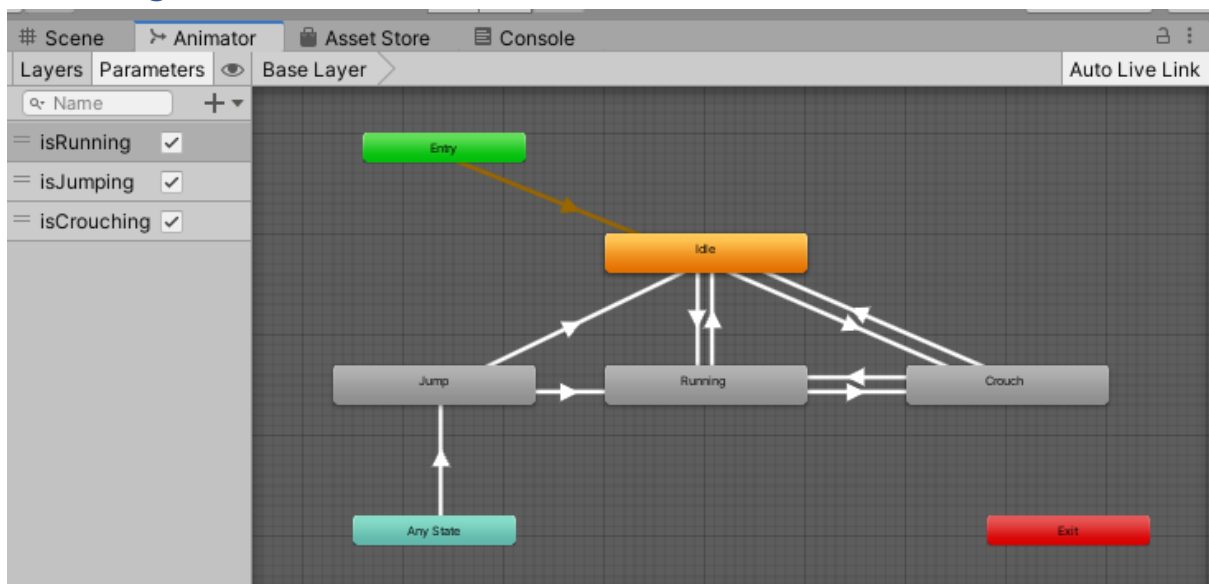
Scripts hooked up with the objects, Main Camera, Player and Enemy:

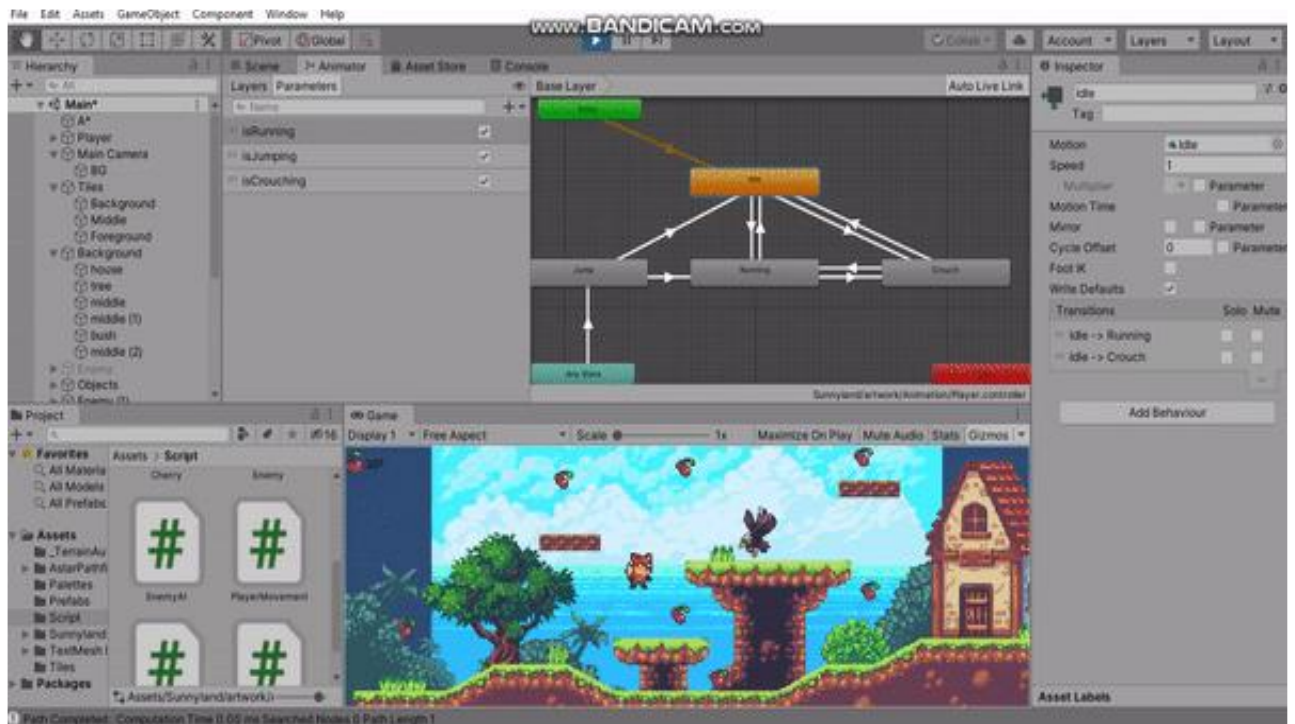
1. Player: CharacterController2D, PlayerMovement and ScreenBoundary
2. Enemy: ScreenBoundary, EnemyAI and Seeker
3. Cherry (GameObject): AutoSpawn
4. Cherry (Prefab): Cherry, ScreenBoundary
5. ScoreManager (GameObject): ScoreManager
6. A* (GameObject): A* script (Pathfinder)

GIF of the Scene Play:



FSM Diagram used for animation:





References used for this game design and implementation:

- ✓ Waldo (n.d.). *Keep Object In Bounds*. [online] [presstart.vip](https://presstart.vip/tutorials/2018/06/28/41/keep-object-in-bounds.html). Available at: <https://presstart.vip/tutorials/2018/06/28/41/keep-object-in-bounds.html> [Accessed 10 Jun. 2020].
- ✓ Waldo (n.d.). *Spawning Obstacles*. [online] [presstart.vip](https://presstart.vip/tutorials/2018/09/25/58/spawning-obstacles.html). Available at: <https://presstart.vip/tutorials/2018/09/25/58/spawning-obstacles.html> [Accessed 10 Jun. 2020].
- ✓ answers.unity.com. (n.d.). *Implement spawning coins in my 2d game - Unity Answers*. [online] Available at: <https://answers.unity.com/questions/1606735/implement-spawning-coins-in-my-2d-game.html> [Accessed 10 Jun. 2020].
- ✓ Unity - Keeping The Player Within Screen Boundaries. (2018). *YouTube*. Available at: https://www.youtube.com/watch?v=ailbszpt_AI [Accessed 10 Jun. 2020].
- ✓ Technologies, U. (n.d.). *Unity - Manual: Constraints*. [online] [docs.unity3d.com](https://docs.unity3d.com/Manual/Constraints.html). Available at: <https://docs.unity3d.com/Manual/Constraints.html> [Accessed 10 Jun. 2020].
- ✓ 2D Animation in Unity (Tutorial). (2018). *YouTube*. Available at: <https://www.youtube.com/watch?v=hkaysu1Z-N8> [Accessed 23 Oct. 2019].

- ✓ HOLD JUMP KEY TO JUMP HIGHER - 2D PLATFORMER CONTROLLER - UNITY TUTORIAL. (2018). *YouTube*. Available at: <https://www.youtube.com/watch?v=j111eKN8sJw&t=1> [Accessed 10 Jun. 2020].
- ✓ Blackthornprod (2020). *HOW TO MAKE 2D GAME ANIMATIONS IN UNITY - BEGINNER TUTORIAL*. *YouTube*. Available at: <https://www.youtube.com/watch?v=EmbA-AitPow> [Accessed 10 Jun. 2020].
- ✓ Blackthornprod (2020). *HOW TO MAKE ANIMATION TRANSITIONS - UNITY TUTORIAL*. *YouTube*. Available at: https://www.youtube.com/watch?v=HVCsg_62xYw&t=3 [Accessed 10 Jun. 2020].
- ✓ MAKING RUN, IDLE & JUMP 2D GAME ANIMATIONS - UNITY TUTORIAL. (2019). *YouTube*. Available at: <https://www.youtube.com/watch?v=FTxQKHG5WCA> [Accessed 10 Jun. 2020].
- ✓ Unity (2020). *The Animator Controller - Unity Official Tutorials*. *YouTube*. Available at: https://www.youtube.com/watch?time_continue=93&v=JeZkctmoBPw&feature=emb_title [Accessed 10 Jun. 2020].
- ✓ Brackeys (2019). *2D PATHFINDING - Enemy AI in Unity*. *YouTube*. Available at: <https://www.youtube.com/watch?v=jvtFUfJ6CP8>.
- ✓ arongranberg.com. (n.d.). *A* Pathfinding Project*. [online] Available at: <https://arongranberg.com/astar/> [Accessed 10 Jun. 2020].
- ✓ Blackthornprod (2020). *PATROL AI WITH UNITY AND C# - EASY TUTORIAL*. *YouTube*. Available at: <https://www.youtube.com/watch?v=8eWbSN2T8TE> [Accessed 10 Jun. 2020].
- ✓ 2D PLATFORMER PATROL AI WITH UNITY AND C# - EASY TUTORIAL. (2018). *YouTube*. Available at: <https://www.youtube.com/watch?v=aRxuKoJH9Y0>.
- ✓ GitHub. (2020). *Brackeys/2D-Movement*. [online] Available at: <https://github.com/Brackeys/2D-Movement.git>.