The Descent

Team Crypt Crawlers members:

Evan Dyson (Project Manager / Coder),

Alex Fleis (Scrum Master / Coder),

Shawn Banks (Coder),

David Kirkendall (Art/Assets),

Collin Guyer (Art/Assets)

CIS 4914 Senior Project

Faculty Advisor: George (Nick) Heitzman

Faculty Advisor Email: gheitzman@ufl.edu

Table of Contents

Abstract

Our video game is a dungeon/level explorer that gets progressively harder the further you advance. There are different mechanics that are provided to the player at the outset of the game via tools and weapons. Throughout the levels, we slowly introduce new enemies and traps that must be defeated or avoided. The abilities and items granted to the player must be utilized to get past these enemies and traps to reach the end. Hidden within levels we planned to have hidden passages and items that can grant a greater boost to the player, but sadly were unable to complete this due to time constraints.

Keywords

•Software and its engineering ~ Software notations and tools ~ Software libraries and repositories

•Theory of computation ~ Theory and algorithms for application domains ~ Algorithmic game theory and mechanism design ~ Algorithmic game theory

•Theory of computation ~ Theory and algorithms for application domains ~ Algorithmic game theory and mechanism design ~ Algorithmic mechanism design

•Theory of computation ~ Theory and algorithms for application domains~Algorithmic game theory and mechanism design ~ Solution concepts in game theory

• Theory of computation ~ Theory and algorithms for application domains ~ Algorithmic game theory and mechanism design ~ Convergence and learning in games

• Computing methodologies ~ Machine learning ~ Machine learning approaches ~ Stochastic games

Introduction

Problem Statement

In the realm of software engineering and game development, creating engaging and challenging gameplay experiences is a constant pursuit. Balancing difficulty progression, introducing new mechanics, and keeping the player engaged are essential challenges in designing a successful game. The project aims to address these challenges by creating a dungeon/level explorer game that offers progressively challenging levels, introduces new mechanics and enemies, and rewards strategic thinking and exploration. By incorporating these elements, the game seeks to provide an immersive and dynamic experience for players. This work is significant in the relevant literature as it contributes to the understanding of game design principles and techniques for creating compelling gameplay experiences. It aligns with existing research on player engagement, level design, and game mechanics, offering practical insights into how these elements can be effectively implemented to enhance player enjoyment and immersion. Additionally, by focusing on the practical application of these concepts in a specific game project, this work provides a tangible example of how theoretical principles can be translated into real-world solutions within the field of game development.

Related Work

Our team took a lot of inspiration from many of the games that we play. During the project showcase while demoing our game for visitors, it was said to be similar to Mario games, one of the genres that we talked about during development many times. In addition to the AAA games that many of us are used to, we also took inspiration from many indie game developers and self-made developers who showcase their skills and offer in-depth tutorials and detailed videos on YouTube. Our solution to the issue of creating a unique and cultivating video game was to incorporate as many features that we could manage that would add to the story and gameplay of our video game.

Solution Statement and Contribution

The proposed solution is to develop a game using Unity and C# scripts, leveraging the capabilities of both platforms to create an engaging and challenging gameplay experience. The solution builds upon and extends current technology by integrating advanced game development tools and techniques to enhance experience for the player. Using Unity’s game engine capabilities, the project implements different level designs, interactive items, and an array of obstacles to overcome. Additionally, the use of C# scripting will allow for the implementation of complex game mechanics and character interactions, adding depth and challenge to the gameplay.

Public Domain

The problem addressed lies within the realm of game development which is a branch of computer science and engineering that focuses on creating interactive digital experiences. Specifically, the problem revolves around designing engaging gameplay experiences that balance difficulty progression, introduce new mechanics, and keep players interested. Our solution fits into this area by leveraging Unity and C# to develop a dungeon explorer game that offers challenging levels, gradually introduces new mechanics and enemies, and rewards strategic thinking and exploration. The solution integrates principles of game design, software development, and user experience to advance the idea of creating an engaging gaming experience.

Literature Review

The major focus of this project, as stated before, is to allow each group member to expand and practice their skills with development using C# and the Unity game engine. Moreover, a significant element of successful game design is striking a balance between providing the player with appropriately interesting and detailed challenges while not hampering the user experience or the flow of the game by inappropriately scaling difficulty. It was thus important to understand the psychology of video games and how their content can affect or be affected by a user. To ensure the best possible end result is delivered, this group decided to conduct research into how the flow and progression of difficulty and the addition of player stress can affect a user's responses to and engagement in the game, and how existing emotions may change a user's experience with a game. This was all in an effort to make the end product as well-rounded and versatile to every type of player possible.

In a 2013 report from Michael W. Buncher from Cleveland State University titled *The Effects of Video Game Difficulty Selection on Flow Experience*, in an effort to test how emotion and level difficulty selection and performance are connected, groups of participants were pre-conditioned to experience the emotions of boredom or stress through activities of each respective nature for 20 minutes, then were asked to play the game Tetris for 15 minutes while a research team took down their scores. The bored and stressed participants were not informed that their Tetris gameplay would be part of the current study, and instead were told it was for an unrelated study so as to achieve more unbiased results. These scores were tested against a control group of non-emotionally influenced participants (Buncher, 2013). The team predicted that stressed players would end up choosing easier levels and play much slower than did their bored or normal counterparts. They also expected bored players to select more challenging levels and play much quicker than the other two groups. The research team discovered that most of their predictions were ultimately right, but the results they had expected were much more subtle and nuanced in nature. Stressed players did indeed select slower levels and play slower in the first section of the trial, but these results began to show less and less deviation in later parts. For bored players, in the first half of the trial the team did not see the expected trend; in fact, bored players tended towards easier level selection and slower gameplay styles, playing faster only at the very final section of the trial. In regards to gameplay flow, neither of the three groups displayed any significant indication of hampered gameplay flow due to their respective emotions (Buncher, 2013). For the purposes of the game this group intends to create, this report and its findings have exemplified the need for greater diversity in both level structure and difficulty to provide an adequate gameplay flow. Once a user makes it through their initial difficulty selection their participation and perception of gameplay flow seem to even out regardless of emotions, so providing a varied starting point is important.

A separate article from 2011 titled *Measuring the level of difficulty in single player video games* highlights important characteristics and a definition for both challenge and difficulty, respectively, through a synthesis of other scholarly sources. The article highlights the importance of difficulty as a driving force for a player's connection to the game, stating that, "the fact that a player exerts effort… and feels attached to the outcome is the core point (Aponte et. al., 2011)." Difficulty thus becomes an integral part of what draws a player into the game, what makes it worth playing, and in the eyes of this group, what makes having a strong sense of balanced difficulty incredibly important to the final product. The article further separates challenge and difficulty, stating that providing users with challenge is a core part of game flow and is in fact one of its key pillars. The article brings in references to Ralph Koster and his book *A Theory of*

Fun for Game Design, in which he describes these challenges as a means to discover patterns and strategies that elevate a player's enjoyment of a game (Aponte et. al., 2011). The article frames these challenges as being ways to not only amp up difficulty overall, but provide methods and patterns to alleviate some sources of difficulty along the way. This is discussed at length in a section referencing Neil Kirby's work with the game Minesweeper, in which he discovered that the difficulty of the game could be dramatically cut by employing certain strategies to reliably clear the majority of the game board (Aponte et. al., 2011). From this article and its analysis of other related articles this group has determined that to create a successful game, not only must there be some form of challenge to increase difficulty and draw the player in, that difficulty must be framed in such a way that the player can derive meaning and gameplay patterns so as to alleviate that difficulty in some respects. Both of these elements contribute to the overall creation of a successful game.

Solution

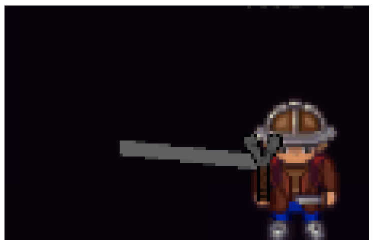
Obstacle 1: Creating the player

Our development of the player began with our movement script. We knew that to have a smooth feeling game the movement script needed to be finished and polished early on in the game. The next thing we focused on was finding the right player model that would fit the feeling of our game and adding in the player animations. After the development of the player was nearing completion we added a way to measure the amount of health the player has left. We did this by attaching a custom-made player health script which holds a lot of the functions of our player such as: Health bar UI control, the slowdown effect from our spider web trap, the respawn system, and the health regenerating. The player also needed a way to respawn when going out of bounds of the game so if the player reached a Y position of <-9 this would cause the player to respawn, additionally, if the enemies were to cause the player's health to be <= 0 it would also cause this same event. After finishing all the player scripts and assets we then focused on the creation of items that our player would be able to use. An example image of the player and its attached objects will be supplied and talked about in the appendices.

Obstacle 2: Creating Player Items

The player created will have three different items to be used in-game: a dagger, a grapple gun, and a slingshot.

- Dagger: a melee weapon that works by having a transform point on the side that the character is facing and by using left click will cause damage to an enemy when it collides with the transform point [7].
- Grapple gun: has a specific layer that allows for grappling, the “ground” layer that we have set to objects that we want the player to be able to grapple to. The grapple gun also has an adjustable length to allow for the player to have more rope while dangling, and a delay between shooting, and it uses the included line renderer script from Unity to render a line that the player shoots out and hangs from.
- Slingshot: works by using the mouse position to determine the direction of the rock projectile [4]. The slingshot is activated when holding right click and increases in damage and range by holding right click. This increase in range and damage is visualized with a force vector as shown below. The slingshot also has a timer element in it that is used for the item cooldown.



Obstacle 3: Puzzle Items

For the puzzle game to work, we needed to have many different items. This included: Spike Traps, Dart trap, Trap door, spider web trap, Dead slime you bounce off of, falling items, ungrappleable points, respawn points, a teleport system, and destructible items.

- Spike trap: has a collider on it that when hit by the player causes the player to respawn.
- Dart trap: has a pressure plate using a trigger that triggers a dart to be spawned in the middle of the emitter and goes in the direction of the pressure plate for flexibility when creating levels. If the player collides with the dart it causes 50 health to be removed.
- Trap door: uses the included Unity "Hinge Joint 2D" script to give the object a rotation point. After setting a rotation point, a maximum and minimum angle can be set to allow for the object to rotate at a specific angle. For our trap door, we used the angles 0° - 90° [9].

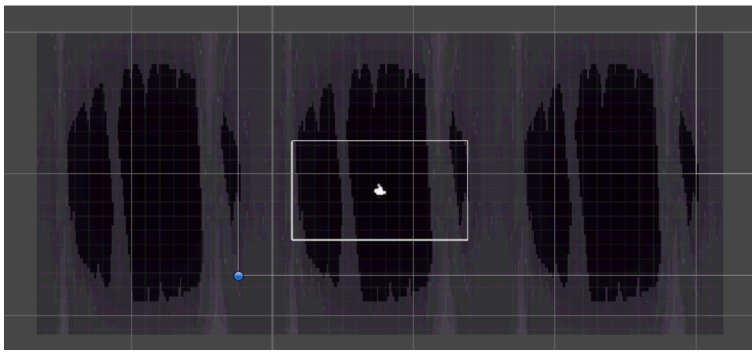
- Spider web trap: spawned from the wall spider enemy when colliding with the player causes the speed at which the player can move to decrease for a few seconds.
- Dead slime (bouncy item): uses Unity's physics system to create a material with a bounciness property that is then applied to a game object to give it bouncy properties.
- Falling objects: having items that are not locked in the y direction makes them be affected by gravity.
- Mossy ground: using Unity's tag system an asset is tagged with "Mossy" and is unable to be grappled to while also creating a physics material with no friction gives this item a slippery feel.
- Respawn points: when entering the collider the respawn point object attached to the player changes to the new transform point and is then disabled in the editor. If being used for the respawn point of the new level a boolean is enabled to reset the player's health.

- Teleport system: causes the player to be moved to the location of the next level or whatever location specified using a transform point, additionally the respawn point is changed and the light level is set to 0 and then increases back to the normal level to act as a transition between levels.
- Destructible objects: using Unity's tag system an item tagged with "Breakable" can be disabled when colliding with the slingshot ammo.

The Teleport system, Trap door, Spike trap, and Dart trap functionalities will be discussed in more depth within the appendices section, as well as including images of how they function.

Obstacle 4: Background

This is accomplished with a parallax script [13] that allows for the distance of each layer from the player to change the rate of speed at which the layer moves. To prevent the background from moving beyond the player's camera there are three copies of the background which will move to whatever side they need so that the player remains in the middle background. This script is used in both the main and boss scenes.



Obstacle 5: Enemies

Within our game, there are 5 different enemies and a boss enemy that the player will encounter while playing. To control our enemies we utilized multiple custom-made scripts that were tailor-made for each enemy. However, to save time and resources these scripts were used for multiple enemies. The enemy health script is attached to all enemies to give them health and upon taking damage causes their enemy sprite to blink red. The enemy damage script is also attached to all enemies, except the red shooting spider, to allow for each enemy to do a unique damage amount to the player.

- Normal spider: has its own script that lets it chase after the player in all directions while also rotating toward the player instead of staying on the ground like the other mobs [5]. While not chasing the player it patrols set points until the player comes within its range.
- Red shooting spider: utilizes a script that makes it stay stationary, rotate towards the player, and shoot a web at the player which causes a slowing effect whenever the player is within range [5, 8].
- Orc: has a variation of the enemy script with one exception, having the player not take damage if the y value of the player is greater than that of its own. Additionally the ogre only patrols and does not seek out the player [2, 3].
- Skeleton: has the original enemy AI script which allows for the enemy to patrol between 2 points, once the player enters within a detection range the enemy then begins to chase after the player [2, 3].
- Slime: also has the original enemy AI script, thus it acts the same way as the skeleton [2, 3].

- Dragon: has a unique dragon AI script, this script controls everything about the dragon. This includes the dragon animations, damage, health/health bar, and movement.

The Spider, Skeleton, and Dragon boss functionalities will be discussed in more depth within the appendices section, as well as including images of how they function.

Obstacle 6: UI

The health is in the top left of the screen and uses Unity's UI system alongside custom sprites. The health bar will also show if the player has an effect like if they are slowed down by the spider. The health bar also will show if any of the player items are on cooldown and displays the key needed to activate the item.

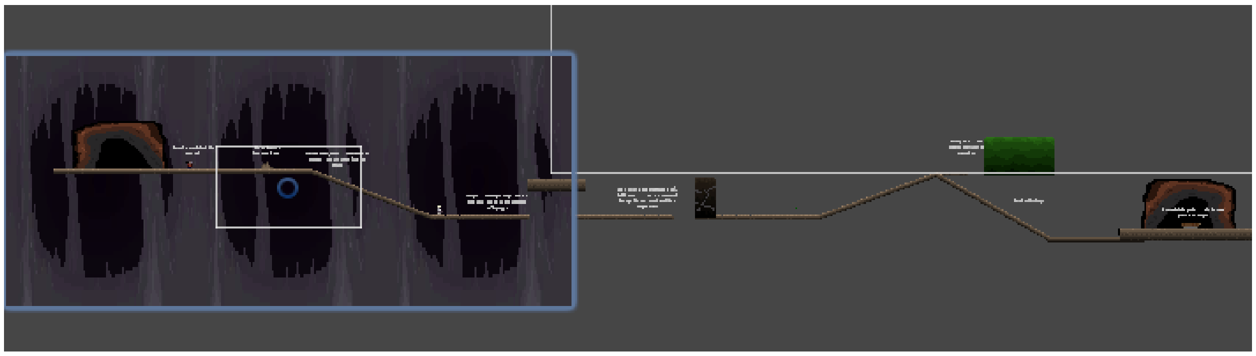
Obstacle 7: Creating the levels

To make sure that we had a variety of level designs for the main game each person on the team created their level on paper to be implemented in the game. As a transition between the levels, the teleport/minecart system is used.

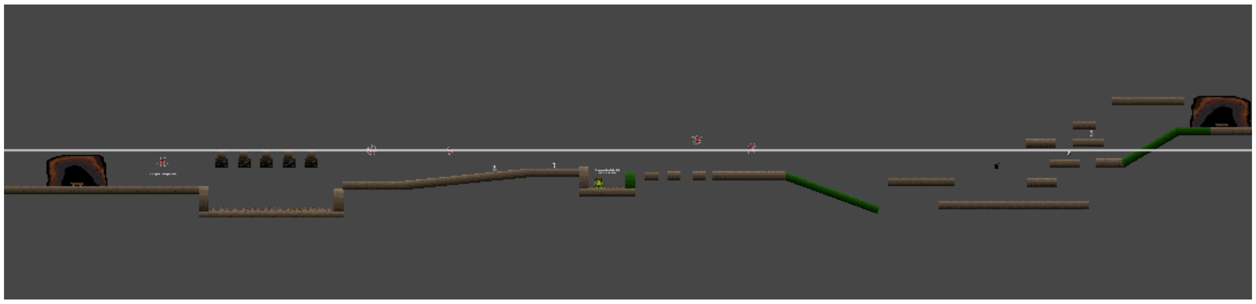
Obstacle 8: Testing

Working in Unity is different from working on a standalone project, one thing that we learned in Unity is that race conditions are something to look out for as things can happen faster or slower than expected causing unexpected issues. Additionally, when creating puzzle games some issues arise in play testing like the ability to cheat and get around a puzzle entirely. With so many parts there are also many test cases that we had not considered until playtesting like behavior of aspects like the grapple gun when the player dies.

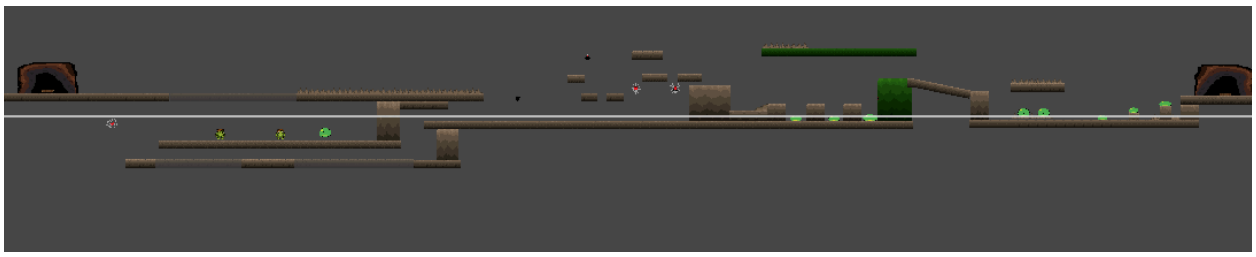
Results



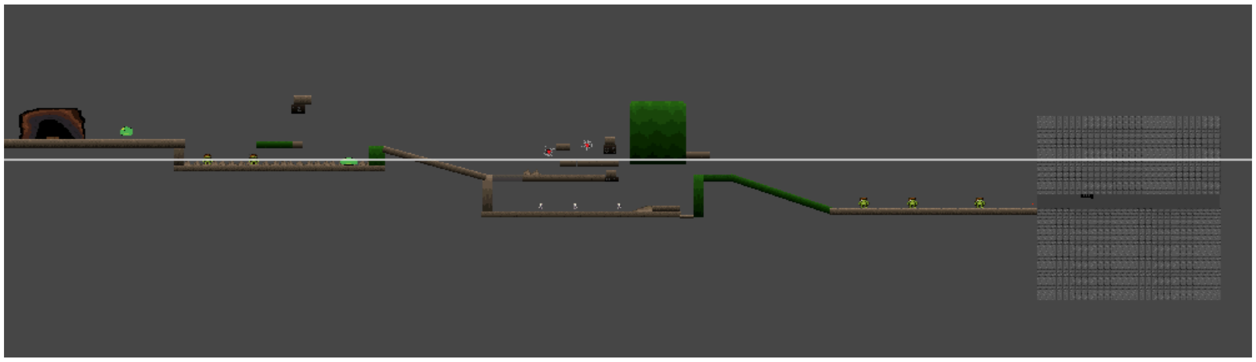
Tutorial Level: Provides instructions on how to play the game



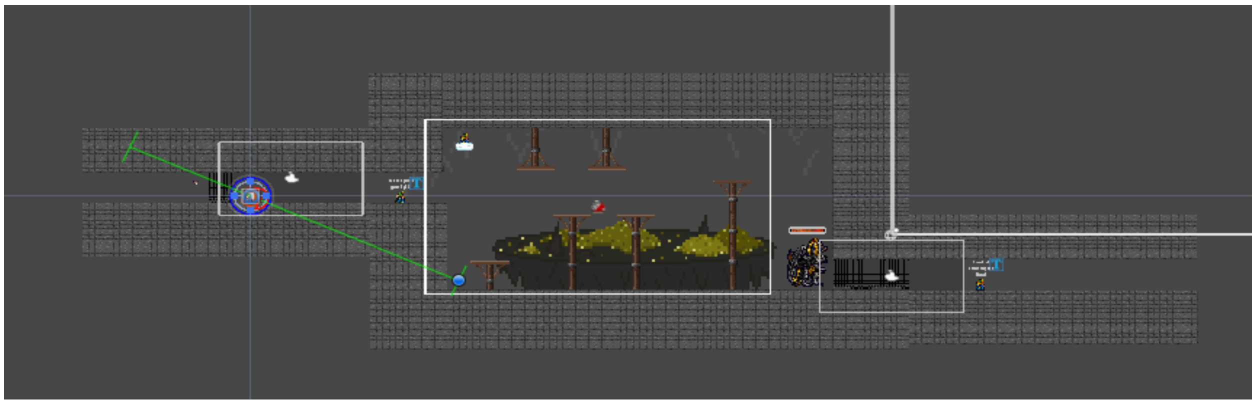
Level 1: Introduces the first puzzle to the game and informs the player about new enemies



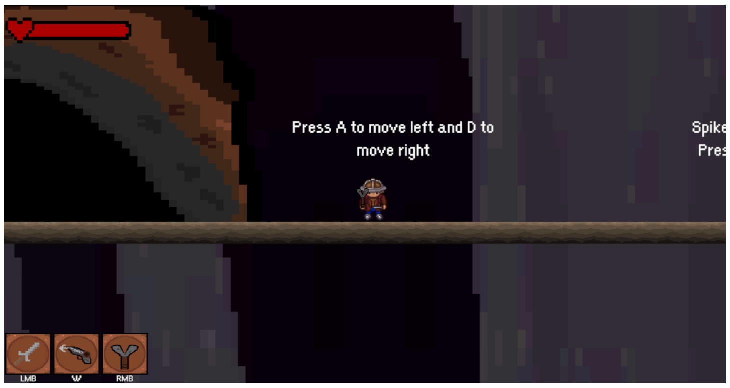
Level 2: More challenging puzzles



Level 3: Last of the puzzle levels and has the entrance for the boss level as depicted on the right



Final Level: Layout of the final boss battle scene



Gameplay screenshot with UI from the main level

Conclusion

In our project, we created a puzzle game full of many different obstacles that allowed for variation in each of our levels as well as providing the player with a challenging experience. Additionally, we created a player that has many different ways to interact with the game world with the melee, ranged, and traversal items. For some of the team, they were less familiar with working in Unity and they were required to quickly learn how to work with it. We learned how to work as a team to create multiple systems that work together and experienced the process of creating any product: pre-production, production, and post-production. In the pre-process phase we came up with concept art and ideas for the systems we wanted to implement. In the production and post-production phases, we learned how to create an interconnected product and how to account for issues that arise when working with these individual parts. For further development of our product, we plan to add more content to the game like new traps and player items as well as improving aspects that we ran out of time to polish like that of the game art. In the development of our project, we learned how to work with C#, the language the Unity scripts are written. This was a new programming experience as Unity works differently than other programming languages with files being written in two basic sections outside of just creating functions: Start() and Update(). The Start() part of the program is used to obtain objects stored outside that script and you access them in Update() as well as any other object in the script. One of the obstacles that we had to overcome when working with Unity and GitHub specifically was that the scene would get corrupted if two people tried to make changes to a scene at the same time. This required us to make a testing scene so that we could still build systems to be implemented into their respective scenes later. Some advantages of our game are that we created what we set out to create, a puzzle adventure game filled with different interactable obstacles and enemies and we had enough time to comprehensively test each level. One of the disadvantages of our product is that we made it a bit too challenging than we realized. Because we were the ones creating and testing our game we didn’t realize until Senior Showcase how hard the game was. Additionally, because our game is linear once someone finishes our game there's little in terms of replayability.

Standards and Constraints

Standards: All programming was developed in Unity 2021.3 using C# version 8.0 and the included .NET functionality of Unity.

Constrains: All software was developed to run in real time, and no assets were unloaded during playtime. Meaning, our whole game was preloaded and ready to play with no loading screens.

Acknowledgments

We would like to dedicate this section to Professor Nick Heitzman. Professor Heitzman provided much-needed insight into art development and for GitHub usage regarding our large gaming files. He also provided guidance for us throughout the semester involving development.

A large part of development involved following tutorials on how to get specific systems interacting with each other properly and also proper implementation. The following list of YouTube channels provided a huge help towards our development. We will include the specific tutorial videos in our reference page.

- Morebblakeyyy
- ChrisTutorialsYT
- Blackthornprod
- NightRunStudio
- ThatOneUnityDev
- TurboMakesGames
- RootGames
- Danidev
- Radiobush

References

[1] J. Descottes. “Piskel: Free online sprite editor.” Piskel. https://www.piskelapp.com/ (Accessed Spring, 2024).

[2] MoreBBlakeyyy. “Simple 2D Enemy Patrolling Unity tutorial.” YouTube. https://www.youtube.com/watch?v=RuvfOl8HhhM&t=332s (Accessed Spring, 2024).

[3] MoreBBlakeyyy. “Unity simple 2D Enemy AI Follow Tutorial.” YouTube. https://www.youtube.com/watch?v=2SXa10ILJms (Accessed Spring, 2024).

[4] MoreBBlakeyyy. “Unity 2D Aim and Shoot at mouse position Tutorial.” YouTube. https://www.youtube.com/watch?v=-bkmPm_Besk&t=300s (Accessed Spring, 2024).

[5] Chris’ Tutorials. “How to Make a Flying Enemy - 2D Platformer Crash Course in Unity 2022 (Part 22).” YouTube. https://www.youtube.com/watch?v=Kz7j-Gh1nZ0&t=1s (Accessed Spring, 2024).

[6] ayockel90, SalahChafai160 and Mars530x. “Sprite facing wrong direction.” Unity Discussions. https://discussions.unity.com/t/sprite-facing-wrong-direction/221901 (Accessed Spring, 2024).

[7] Blackthornprod. “HOW TO MAKE 2D MELEE COMBAT - EASY UNITY TUTORIAL.” YouTube. https://www.youtube.com/watch?v=1QfxdUpVh5I&t=1s (Accessed Spring, 2024).

[8] MoreBBBlakeyyy. “2D Enemy Shooting Unity Tutorial.” YouTube. https://www.youtube.com/watch?v=--u20SaCCow (Accessed Spring, 2024).

[9] Night Run Studio. “Trap Doors and Spinning Death Sticks – Hinge Joints in Unity #1.” YouTube. https://www.youtube.com/watch?v=yGBedTRvlYs (Accessed Spring, 2024).

[10] ThatOneUnityDev. “How To Make Any Game Mechanic - Episode 7 - 2D Grappling Hook.” YouTube. https://www.youtube.com/watch?v=Gx46xUgVXrQ (Accessed Spring, 2024).

[11] Root Games. “Unity Tilemap: Fix Gaps Between Tiles (100% WORKS).” YouTube. https://www.youtube.com/watch?v=pXc-H0pb668&t=1s (Accessed Spring, 2024).

[12] Turbo Makes Games. “How to Repeat Tile Sprites in Unity.” YouTube. https://www.youtube.com/watch?v=ug2v1nSCZSk&t=73s (Accessed Spring, 2024).

[13] Dani. “Unity Parallax Tutorial - How to infinite scrolling background.” YouTube. https://www.youtube.com/watch?v=zit45k6CUMk (Accessed Spring, 2024).

[14] J. French. “How to add a background image in Unity.” gamedevbeginner. https://gamedevbeginner.com/how-to-add-a-background-image-in-unity/ (Accessed Spring, 2024).

[15] New World Computing, KnowWonder and Maxim. “Black Dragon.” The Spriters Resource. https://www.spriters-resource.com/pc_computer/heroesofmightandmagic2/sheet/29291/ (Accessed Spring, 2024).

[16] Square, Square Enix and Tose. “Final Fantasy 5 (JPN) - Playable Characters - Faris Scherwiz.” The Spriters Resource. https://www.spriters-resource.com/fullview/31543/ (Accessed Spring, 2024).

[17] Square and Square Enix. “Common NPC’s.” Sprite Database. https://spritedatabase.net/file/16753 (Accessed Spring, 2024).

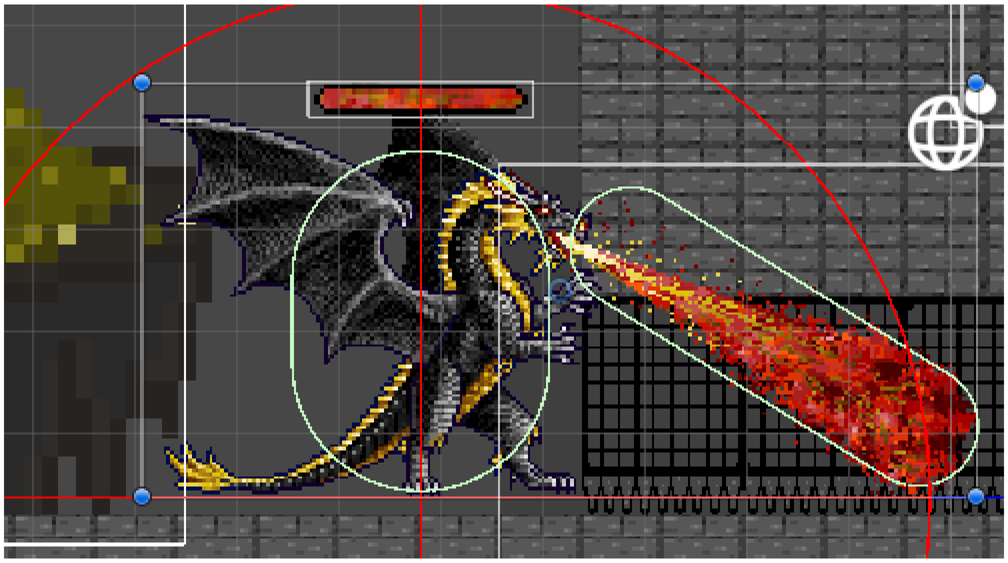
[18] Square, Square Enix and Tose. “Locke.” Sprite Database. https://spritedatabase.net/file/7812 (Accessed Spring, 2024).

[19] 1001 Fonts. “Orange Kid Font.” 1001 Fonts. https://www.1001fonts.com/orange-kid-font.html (Accessed Spring, 2024).

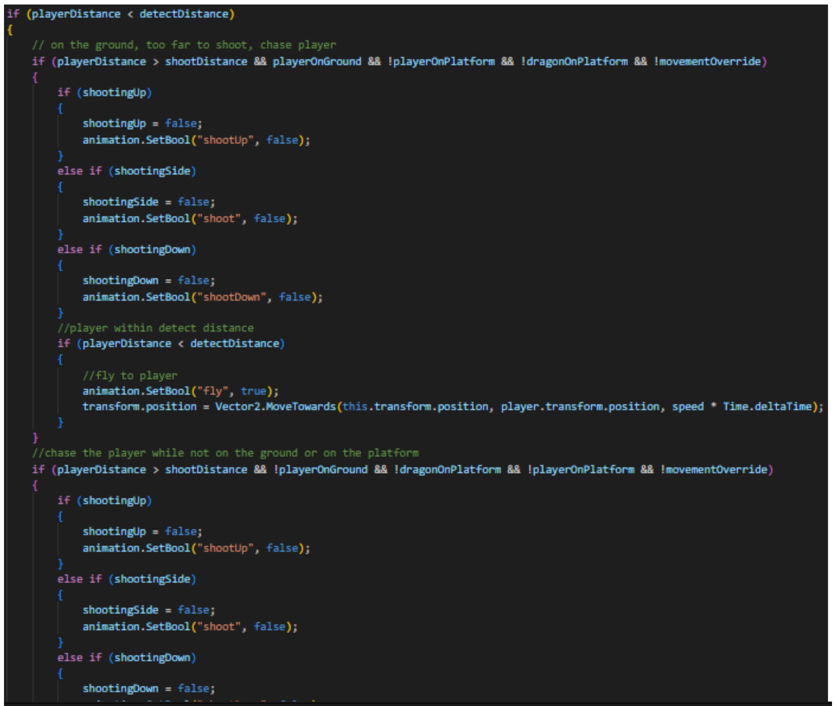
[20] Radiobush. “SceneManager.Loadscene unity code tips c# how to load next scene.” YouTube. https://www.youtube.com/watch?v=bYoNh6T92tM (Accessed Spring, 2024).

[21] Pixabay. “Droplets in a cave.” Pixabay. https://pixabay.com/sound-effects/droplets-in-a-cave-6785/. (Accessed Spring, 2024)

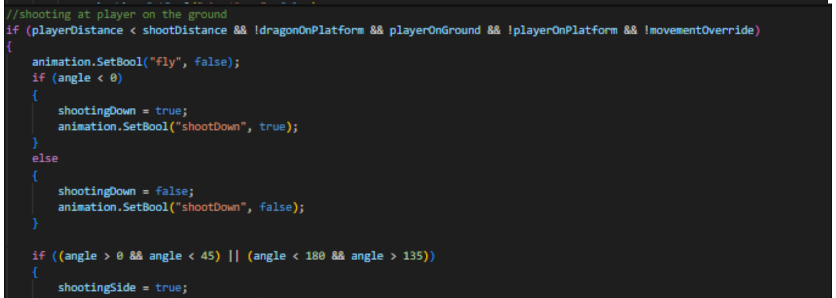
Appendices



To the left the dragon model is shown, along with its collider and flame collider. While the dragon is attacking, a custom traced collider is activated which encases the fire to ensure that if the player is inside of the collider then it will take damage.

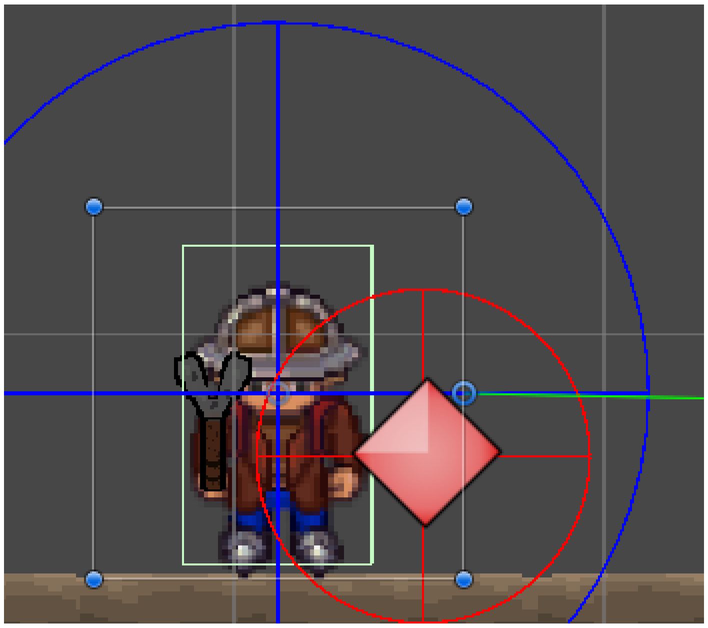


```
if (playerDistance < detectDistance)
{
    // on the ground, too far to shoot, chase player
    if (playerDistance > shootDistance && playerOnGround && !playerOnPlatform && !dragonOnPlatform && !movementOverride)
    {
        if (shootingUp)
        {
            shootingUp = false;
            animation.SetBool("shootUp", false);
        }
        else if (shootingSide)
        {
            shootingSide = false;
            animation.SetBool("shoot", false);
        }
        else if (shootingDown)
        {
            shootingDown = false;
            animation.SetBool("shootDown", false);
        }
        //player within detect distance
        if (playerDistance < detectDistance)
        {
            //fly to player
            animation.SetBool("fly", true);
            transform.position = Vector2.MoveTowards(this.transform.position, player.transform.position, speed * Time.deltaTime);
        }
    }
    //chase the player while not on the ground or on the platform
    if (playerDistance > shootDistance && !playerOnGround && !dragonOnPlatform && !playerOnPlatform && !movementOverride)
    {
        if (shootingUp)
        {
            shootingUp = false;
            animation.SetBool("shootUp", false);
        }
        else if (shootingSide)
        {
            shootingSide = false;
            animation.SetBool("shoot", false);
        }
        else if (shootingDown)
        {
            shootingDown = false;
```

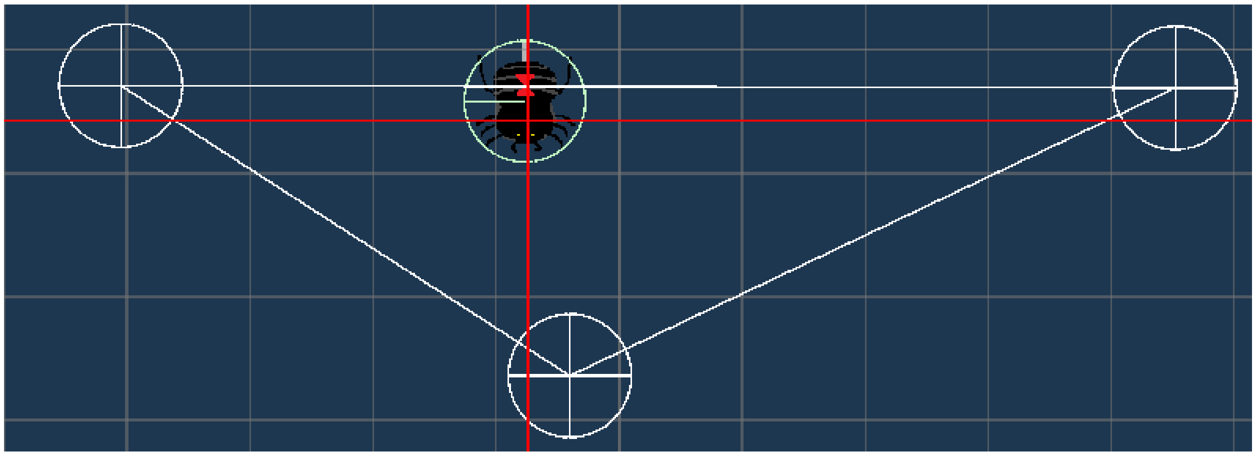


```
//shooting at player on the ground
if (playerDistance < shootDistance && !dragonOnPlatform && playerOnGround && !playerOnPlatform && !movementOverride)
{
    animation.SetBool("fly", false);
    if (angle < 0)
    {
        shootingDown = true;
        animation.SetBool("shootDown", true);
    }
    else
    {
        shootingDown = false;
        animation.SetBool("shootDown", false);
    }

    if ((angle > 0 && angle < 45) || (angle < 180 && angle > 135))
    {
        shootingSide = true;
```

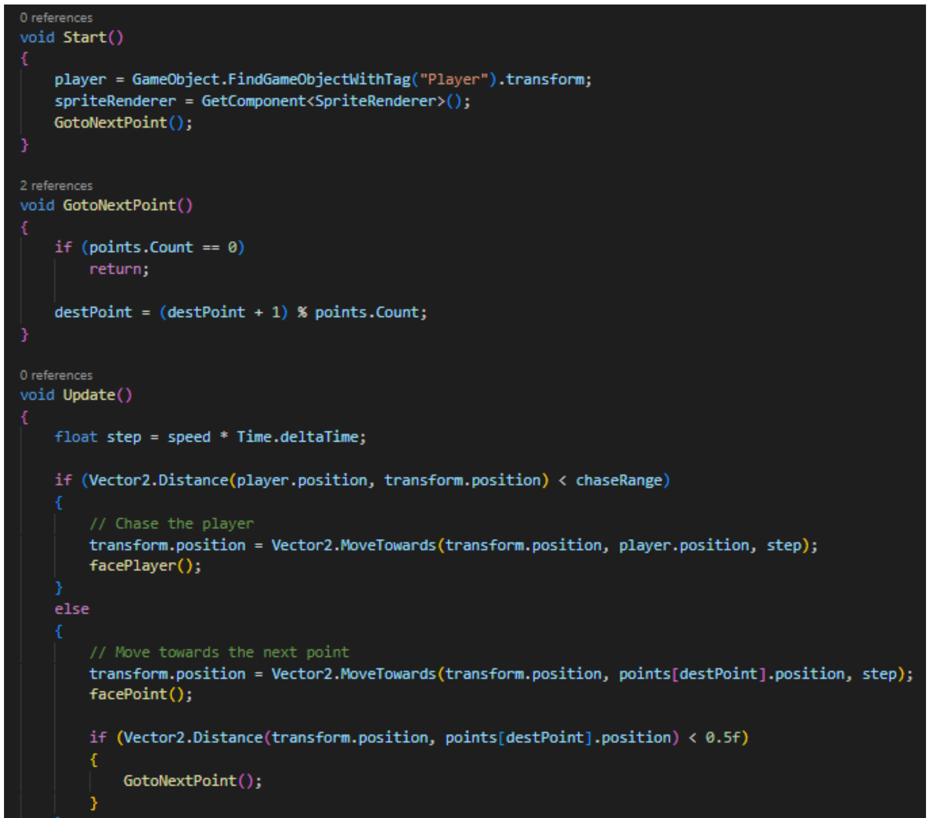
This is the script used to control the dragon. This is only a snippet of the code used, this snippet shows how the dragon movement is controlled. First there is a set detection distance that the player must be inside for the dragon to begin moving towards the player. Next come a set of if statements with multiple booleans to check for what action the dragon should be doing. If the player is not within shooting distance then the dragon flies towards the player. If the player is on the ground and is within shooting distance, then the dragon begins attacking. There are 3 attacks, upwards, downwards, and a normal sideways attack to cover all angles. While shooting fire the dragon detects the angle of the player and changes animation / attack correspondingly. If the player is up on the above platforms, the dragon then flies to a pre-set position and once the dragon reaches its destination it then begins shooting upwards at the player. All while this is happening there is additional code that controls various things about the dragon. There is code for checking to see if the dragon is facing the correct direction by flipping the sprite over the Y axis to ensure that it is always facing the player. There is a health checker to see if the dragon's health is <= 0, if it is then the death animation is triggered and all colliders are turned off as well as the script being deactivated. There is also collision detection to check if the player is touching the dragon and if it is then damage is dealt to the player.



To the left is the player model and some of the ranges and colliders involved. The blue circle shows how far the grapple rope extends after colliding with a ground object. The red circle is the melee collider, this works by detecting if there is an object within the circle when clicking the left mouse button. If there is something within the circle, the tag of the object is checked to determine if the object is an enemy or not, if it is an enemy damage is given to the enemy. Lastly, the light green rectangle that encases the player is the player collider. This makes sure that the player has proper collision with objects within our levels and also with the enemies.

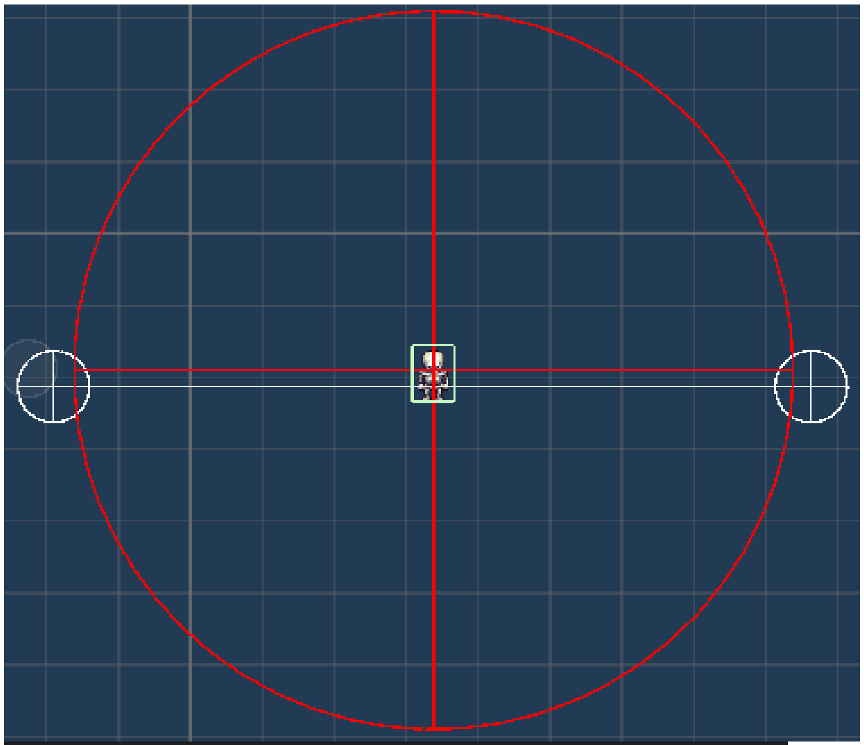


The image above shows off the normal spider enemy. The red lines extend out to a circle which shows the detection range of the spider. The white circles and lines depict the pathing of the spider, while the player is not within range the spider patrols around the path. Finally, the light green circle around the spider is the collider, this ensures for proper collision with the player and allows for it to take damage.

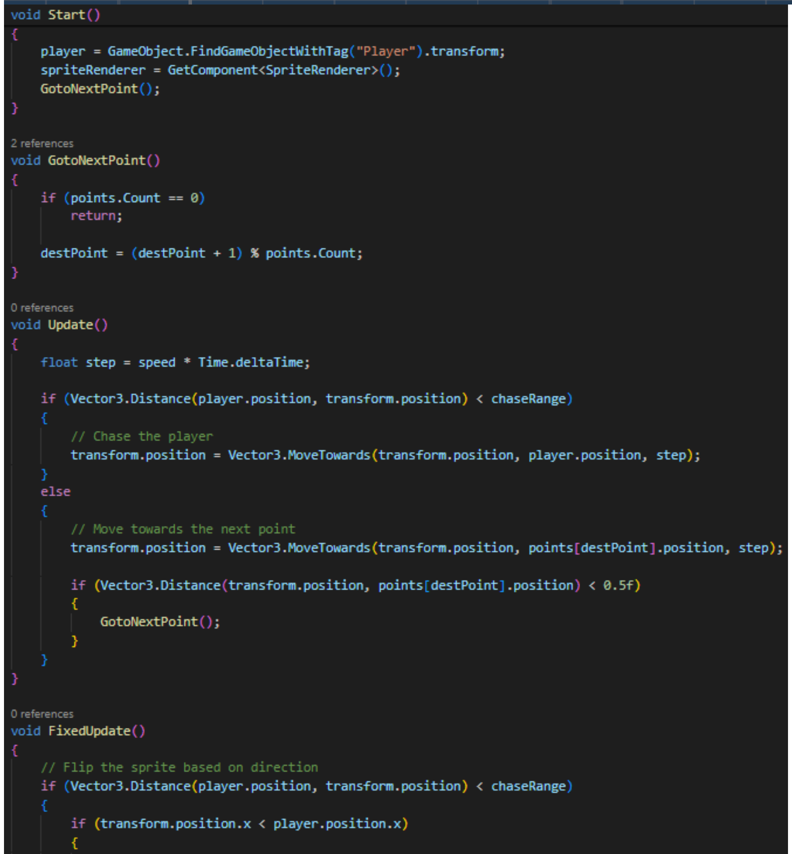


```
0 references
void Start()
{
    player = GameObject.FindGameObjectWithTag("Player").transform;
    spriteRenderer = GetComponent<SpriteRenderer>();
    GotoNextPoint();
}

2 references
void GotoNextPoint()
{
    if (points.Count == 0)
        return;

    destPoint = (destPoint + 1) % points.Count;
}

0 references
void Update()
{
    float step = speed * Time.deltaTime;

    if (Vector2.Distance(player.position, transform.position) < chaseRange)
    {
        // Chase the player
        transform.position = Vector2.MoveTowards(transform.position, player.position, step);
        facePlayer();
    }
    else
    {
        // Move towards the next point
        transform.position = Vector2.MoveTowards(transform.position, points[destPoint].position, step);
        facePoint();

        if (Vector2.Distance(transform.position, points[destPoint].position) < 0.5f)
        {
            GotoNextPoint();
        }
```

To the left, the spider AI is shown. The normal enemy AI script is very similar to this, with the exception that the normal enemies do not need to rotate since they only move horizontally. The script starts by going to the first designated point. If at any time the player enters within the detection range the enemy then begins moving towards the player. If the player exits the detection range the enemy goes back to traveling towards the unreached point. All while moving, the script automatically faces the enemy in the correct direction.

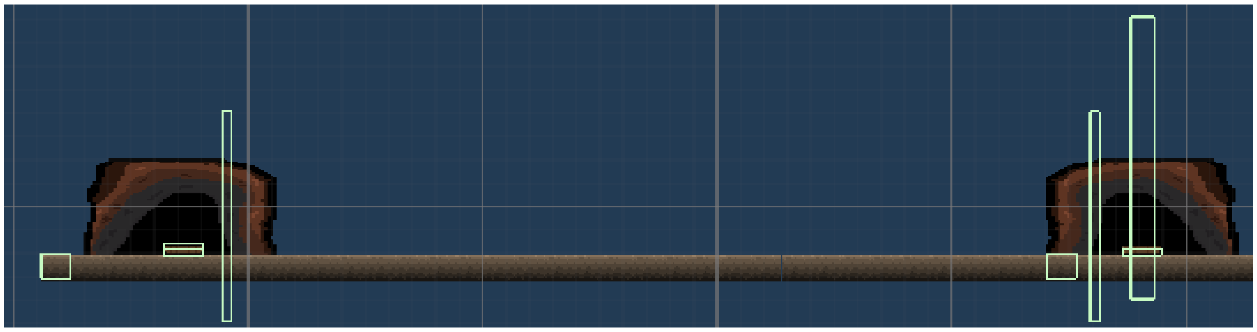


To the left the skeleton enemy is shown. Again, the red circle is its detection range, the white line and circle depict the pathing, and there is a collider attached to the skeleton to ensure proper collision and damage detections.

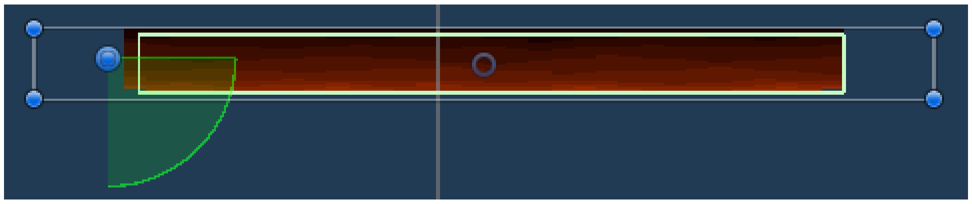


```
void Start()
{
    player = GameObject.FindGameObjectWithTag("Player").transform;
    spriteRenderer = GetComponent<SpriteRenderer>();
    GotoNextPoint();
}

2 references
void GotoNextPoint()
{
    if (points.Count == 0)
        return;

    destPoint = (destPoint + 1) % points.Count;
}

0 references
void Update()
{
    float step = speed * Time.deltaTime;

    if (Vector3.Distance(player.position, transform.position) < chaseRange)
    {
        // Chase the player
        transform.position = Vector3.MoveTowards(transform.position, player.position, step);
    }
    else
    {
        // Move towards the next point
        transform.position = Vector3.MoveTowards(transform.position, points[destPoint].position, step);

        if (Vector3.Distance(transform.position, points[destPoint].position) < 0.5f)
        {
            GotoNextPoint();
        }
    }
}

0 references
void FixedUpdate()
{
    // Flip the sprite based on direction
    if (Vector3.Distance(player.position, transform.position) < chaseRange)
    {
        if (transform.position.x < player.position.x)
        {
```

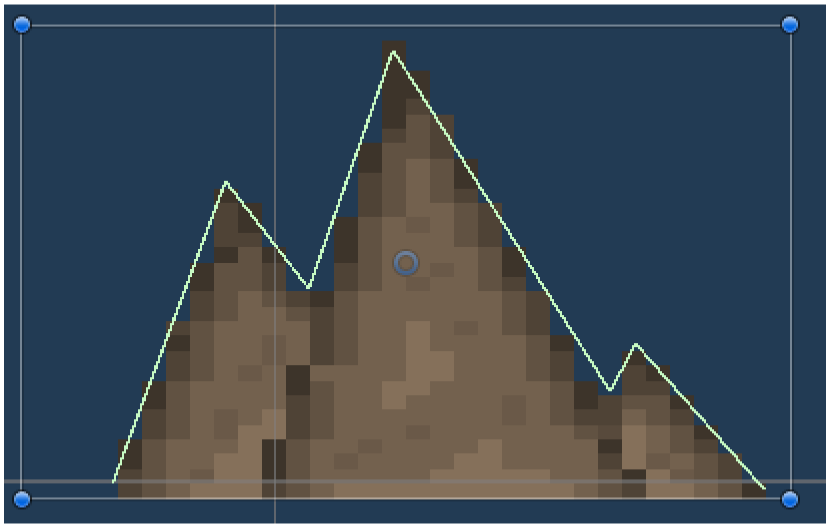
As mentioned above the script used for controlling the skeleton is very similar to the spider. The main difference being that the skeleton travels horizontally so a rotation angle does not need to be calculated to rotate the sprite. A simple flip on the Y axis allows for the enemy to face the correct direction.



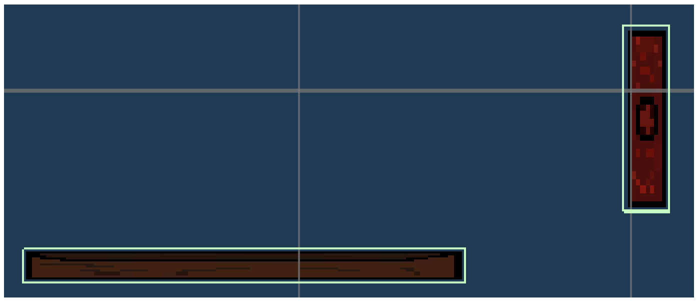
The level changer system as depicted above uses quite a few different systems. On the left in the middle of the cave entrance is a teleporter and upon pressing “T” teleports you to a new transform point which in this case is the middle of the cave on the right side. When teleporting the player's health is reset to 100 and all status effects are removed. Additionally, upon using the level changer the light level of the scene is changed to 0 and then increased to give the illusion of coming in and out on the other side. The long rectangle on the right side in the middle of the cave utilizes the checkpoint system and upon the teleportation of the player will change the player's respawn point to be inside the cave. The skinny rectangles on both caves are colliders to prevent the player from going out of bounds.



This is an image showcasing the trap door[9]. The green semicircle shows the rotation limits of the trapdoor, the sprite and light green collider are only able to rotate along this semicircle limit. Attached to the trap door is Unity’s included “Hinge Joint 2D” script. Within this script developers can set a rotation angle, and a motor. For our trap door, both were needed. The rotation angle of the trap door was from 0° to 90° shown with the green semicircle. The motor was utilized to make the trap door close after being pushed open to give a trap-like feel to the trap door. A speed and force were set to numbers that gave it a natural feel.

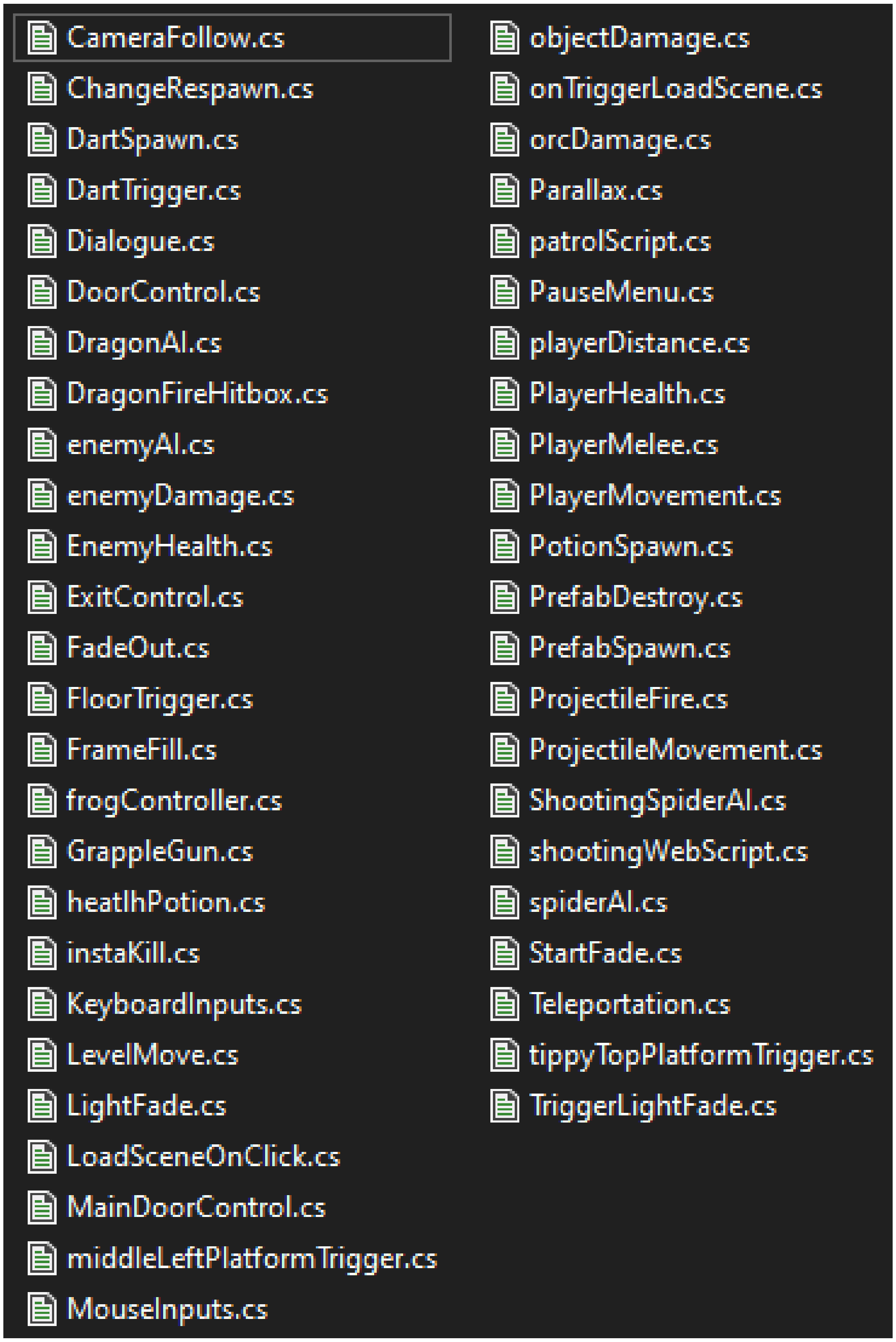


The spike trap uses a polygon collider to closely match the shape of the sprite. If the player touches this it causes the player's health to decrease by 100 and respawn at the set respawn point. This is done using the enemy damage script with a damage value of 100.



To the left is the picture of the dart trap. The dart trap consists of two parts: the pressure plate on the left and the emitter on the right. When the player collides with the pressure plate it spawns a prefab of a dart in the center of the emitter. The dart is a prefab that uses the enemy damage to deal 50 damage to the player. The pressure plate can be placed on either side of the emitter and it will shoot in the direction of the pressure plate.

This image on the next page showcases all of the custom scripts that were used within our game.



```
CameraFollow.cs              objectDamage.cs
ChangeRespawn.cs             onTriggerLoadScene.cs
DartSpawn.cs                 orcDamage.cs
DartTrigger.cs               Parallax.cs
Dialogue.cs                  patrolScript.cs
DoorControl.cs               PauseMenu.cs
DragonAI.cs                  playerDistance.cs
DragonFireHitbox.cs          PlayerHealth.cs
enemyAI.cs                   PlayerMelee.cs
enemyDamage.cs               PlayerMovement.cs
EnemyHealth.cs               PotionSpawn.cs
ExitControl.cs               PrefabDestroy.cs
FadeOut.cs                   PrefabSpawn.cs
FloorTrigger.cs              ProjectileFire.cs
FrameFill.cs                 ProjectileMovement.cs
frogController.cs            ShootingSpiderAI.cs
GrappleGun.cs                shootingWebScript.cs
heatlhPotion.cs              spiderAI.cs
instaKill.cs                 StartFade.cs
KeyboardInputs.cs            Teleportation.cs
LevelMove.cs                 tippyTopPlatformTrigger.cs
LightFade.cs                 TriggerLightFade.cs
LoadSceneOnClick.cs
MainDoorControl.cs
middleLeftPlatformTrigger.cs
MouseInputs.cs
```

Team Biography

Evan Dyson - I am in my senior year at the University of Florida and graduation date is set for December of 2024. I have been employed with Storage Asset Management for the past 3 years, and I have accepted an internship with Modus Operandi for this summer of 2024. My goal after graduation is to stay employed with Modus Operandi and to continue to expand my skills in software development and engineering. Outside of my schoolwork and job, I enjoy playing with my many animals (6 total), video games, and spending time with my friends and family.

Alex Fleis - I am a senior planning to graduate in spring 2024 from the University of Florida. Two summers ago I completed my first internship at OneTouch Robotics. Last summer I interned at Northrop Grumman and received a return offer which I will be starting following my graduation. Outside of school and work, I enjoy hanging out with friends, hiking, and watching movies.

Shawn Banks - I am a senior planning to graduate this Spring 2024 semester from the University of Florida. I have completed a minor in Digital Arts and Sciences, and plan to pursue a career in software development or cybersecurity. This year I was also a Captain for the UF Archery Club, which was an extremely rewarding experience. I enjoy archery, playing and modifying games, hiking, and other outdoor activities, as they are a good way to reconnect with nature.

David Kirkendall - I am currently a senior at the University of Florida, preparing to graduate in the Spring of 2024 with a degree in Digital Arts and Sciences from the Herbert Wertheim

College of Engineering. Following graduation, I am seeking employment for either a game development company or a large engineering company with a simulation software division. I have spent the last three years with the Streetlight program at the UFHealth Shands Hospital, one of which was spent as a volunteer and the remaining two as a Technology Manager. I have also worked for the Code Ninjas company as an instructor. I have a personal interest in creating story and world building, as I'm very invested in the creative side of projects. I also have an interest in VR technology and its potential applications in both gaming and professional environments. Outside of academics and professional experience I enjoy playing, critiquing and designing video games, both collaboratively and on my own; I also play competitive paintball and enjoy bouldering when I can find a studio.

Collin Guyer - I am a 22-year-old graduate from the University of Florida, expecting to complete my degree in the spring of 2024. My professional journey began in a non-technical role at Amazon, where I honed valuable skills such as teamwork, communication, and punctuality, which I believe will be instrumental in my future career. My primary goal after graduation is to further my knowledge by obtaining certifications in project management or cybersecurity and securing a well-paying job in the tech industry. Outside of work and academics, I enjoy playing baseball, video games, and spending time with friends. These interests provide balance in my life and inspire creativity in everything I do.