COLOURVISION

PORTFOLIO

DES203 | Interactive Media Production

Liam Rickman

1902527@uad.ac.uk

Game Design and Production

Contents

This portfolio showcases my work throughout DES203 in my roles as Lead Programmer and Level Designer.

As I was the Lead Programmer most of my portfolio takes place in scripts. Screenshots have been taken of some key areas of code, however the fully commented versions can be found linked at the bottom of each page as well as at the end of the portfolio. These scripts go into further detail about how each script functions.

A mechanic showcase is available on YouTube at https://www.youtube.com/watch?v=lzxFqtqxteU. This is linked to throughout the portfolio at the bottom of the page.

Concept

ColourVision

ColourVision is a 2D isometric dungeon crawler featuring a lizard wizard who has stolen all the colours from the world leaving it in greyscale. The players main goal is to bring back the colour by defeating the lizard wizard and his many minions while progressing through each level.

It was inspired by television evolution and the Marvel TV show WandaVision, mimicking the 50s to 80s eras of television moving from black and white to coloured.

Three levels were originally designed with a unique boss showing up at the end of each level. A different colour scheme would be associated with each level with enemies taking the colour of the one the player is trying to collect.

For our vertical slice we first planned to create only the final level and the final lizard wizard boss. This was later changed to include the first level as well as this would show of more of the games key mechanic of collecting colour and gradually returning the world to colour.

The two levels we took to design further were:

- Diner: Set in the 50s there is no colour present and the player begins by collecting pink from the enemies.
- Arcade: Set in the 80s, the player has collected pink and cyan so far with the world looking more like normal. They are now trying to collect yellow. The lizard wizard boss will also show up at the end of the arcade level with a boss fight.

Team Roles

Liam Rickman: Lead Programmer / Level Designer

- As lead programmer I was in charge of the majority of the games programming and implementation of assets into the game engine of choice (Unity). I also helped out with level design work in-engine.

Rhiannon Nash: Producer / Level Designer / Design Concepting

- Rhiannon was our producer and kept track of the project as a whole. She also designed the two playable levels and implemented the menu systems.

Rowan James: Lead Designer / Narrative Designer / Design Concepting

- Rowan was our lead designer and narrative designer and made sure the project stuck to his original design idea. He also created various UI art for the menu systems

Amy Coates-Walker: Lead Artist / Concept Artist / Character and Prop Artist

- Amy was the lead artist and ensured all artists created artwork that fit the overall theme and art style. She also created enemy sprites with animations and helped with some player animations.

Harry Scorgie: Sound Effects Designer / AI and Tools Programmer

- Harry helped out with programming different enemy controllers and implemented sound effects and music he created.

Hannah Ross: Environment Artist / Music Designer / Concept Artist

- Hannah was the environment artist and created all the environment art for both levels.

Annie Voikok: Narrative Designer / Concept Artist / Character Artist

- Annie created most of the player sprites including animations. She also created various illustrations for the menu systems

GitHub and Unity Collaborate

GitHub

The first thing I began looking into for programming was how to share the project remotely between the group. I initially began by looking into GitHub which I had used before. I set up the repository and used GitKraken to clone and update the project.

This worked for the purpose we needed however was difficult to get used to especially for those new to GitHub so I decided to look further at other methods.

Unity Collaborate

I was recommended unity collaborate by another classmate so began looking into it. It appeared to be much simpler than GitHub and was built into Unity itself. The only limitation with this was that the free version had a user limit of three, however this was not an issue in the end with only three of us working on the unity project itself.

This is what we kept using throughout the project and was very valuable. It allowed us to all work on the project at the same time with minimal issues. It also showed who was working on particular sections at a time to limit duplicate work.

Tilemaps

Having never worked with an isometric viewpoint, I decided to first look into creating isometric tilemaps.

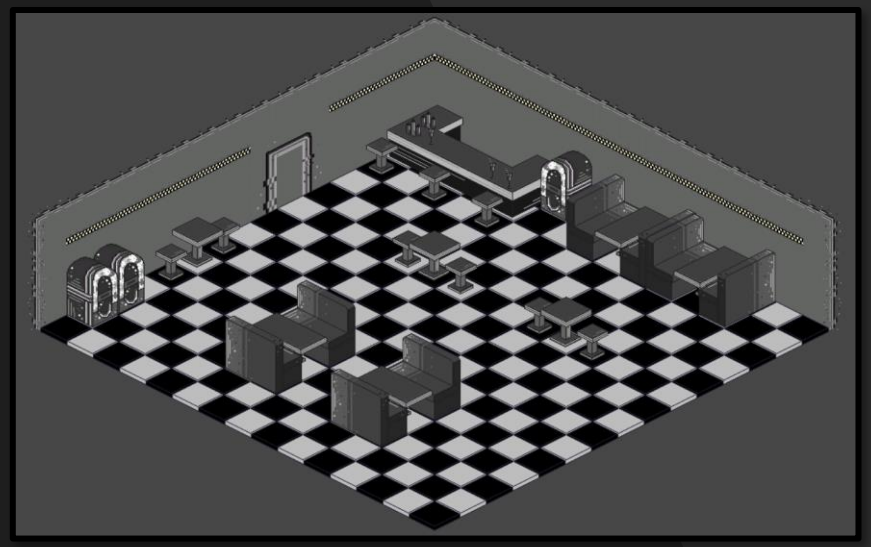
Using a blog on the Unity website *(Hinton-Jones, 2019)* alongside a YouTube tutorial *(samyam, 2020)* I created a basic tilemap using some placeholder assets from the Unity Store that could be used to create level designs while artwork was completed.

Later on these tilemaps were recreated with the new artwork as it was developed. I created a set of tilemaps per level, splitting each into main environment, wall details and furniture.

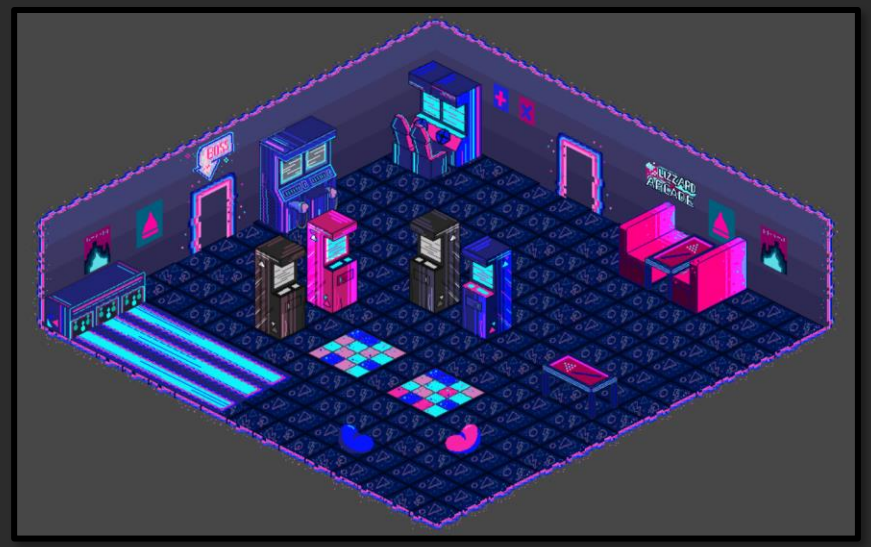
A collision tilemap was also created that was used to manually draw the collisions for each level. I chose to draw collisions manually to give us more control over how collisions functioned in the isometric viewpoint.

I also converted many of these assets to a size format more appropriate for unity and to keep them all consistent. As I knew more about the appropriate sizes than the artists I resized these

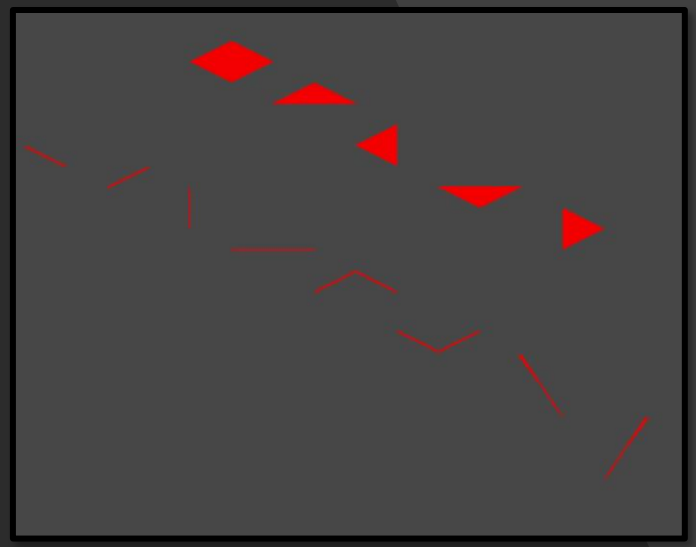
Diner



Arcade



Collisions



Player (Movement)

One of the first parts of scripting I did was the player movement. The isometric view uses a top down movement system for the most part but I had to adapt it slightly to fit the isometric grid.

A state machine was used to swap the movement between standard and a dashed movement shown later.

The script gathers the players horizontal and vertical inputs, converts the Y value to fit the isometric grid and stores both in a vector. This is normalised to ensure the player stays at a constant speed when moving in each direction.

The rigidbody is updated instead of the transform object to make it more consistent and smoother overall.

Gather inputs and calculate move direction

```
//Switch states depending on whether player is moving normally or dashing.
switch (state)
{
    //Standard movement state.
    case State.Normal:

        //Get move directions from input.
        float moveX = Input.GetAxisRaw("Horizontal");
        float moveY = Input.GetAxisRaw("Vertical");

        //Edits move Y speed to fit isometric format.
        moveY = moveY / 2;

        //Set move direction
        moveDirection = new Vector3(moveX, moveY);

        //Normalize move direction so speed is consistant across all directions.
        moveDirection = moveDirection.normalized;
```

Update rigidbody velocity to standard movement

```
//Moves the player correctly depending on what state they are currently in
1 reference
private void MovePlayer()
{
    //Check which movement state the player is currently in.
    switch (state)
    {
        //If in standard movement
        case State.Normal:
            //Sets the rigidbody velocity to the move direction and movespeed.
            rb.velocity = moveDirection * moveSpeed;
            //Allows the player to be damaged while moving normally.
            canDamage = true;

            break;
```

Showcase (0:07): https://youtu.be/lzxFqtqxteU?t=7

Player Controller Script: https://pastebin.com/KqrCVG71

Player (Dash Movement)

The other state the player can move in is a dashing state. In this state the player cannot take damage and moves faster in their current direction.

A YouTube tutorial *(Code Monkey, 2020)* was used to help with the dash calculations.

The dash speed drops slowly over time to gradually bring the player back to their standard move speed once the dash has finished.

As seen in the YouTube video, the dash has a cooldown so it cannot be rapidly pressed and break the game. The player also cannot bypass the collisions and dash through them.

Updates rigidbody velocity to move player

```
//If in dashing movement
case State.Dashing:
    //Sets the rigidbody velocity to the dash direction and speed.
    rb.velocity = dashDirection * dashSpeed;
    //Stops the player from taking damage while dashing.
    canDamage = false;

    break;
```

Calculates dash movement

```
//Dashing movement state.
//Dash movement used a YouTube tutorial to help with the programming: https://youtu.be/Bf_5qIt9Gr8
case State.Dashing:
    //Slows the current dash speed over time.
    currentDashSpeed -= dashSpeed * rollSpeedDrop * Time.deltaTime;

    //Once the dash speed reaches the minimum speed the state is changed back to normal and dash speed is reset.
    if (currentDashSpeed < rollSpeedMinimum)
    {
        state = State.Normal;
        currentDashSpeed = dashSpeed;
    }
    break;
```

Showcase (0:16): https://youtu.be/lzxFqtqxteU?t=16

Player Controller Script: https://pastebin.com/KqrCVG71

Player (Attack)

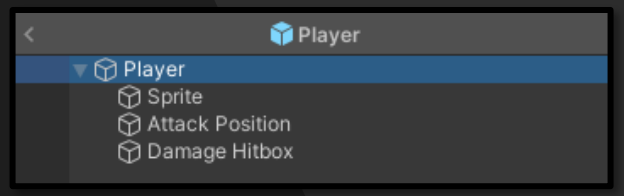
The player attack works by creating a circle collider array around the player that checks for any enemy game objects in range. The circle collider is placed on the attack position in the player prefab and can be repositioned if needed to face one side of the player.

If I had extra time I would go further and make the attack collider match up with the baseball bat so the player can only attack one side at a time.

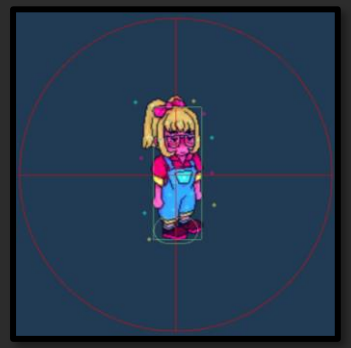
Attack (PlayerAttack)

```
Unity Message | 0 references
private void Update()
{
    //Counts down the attack delay constantly to see if the player can attack
    currentAttackDelay -= Time.deltaTime;

    //Checks if the player is alive
    if(player.GetIsDead() == false)
    {
        //If the attack delay reaches 0 and the player has pressed mouse 1 the player will attack.
        //This will last as long as the player holds down mouse 1
        if (Input.GetMouseButtonDown(0))
        {
            if (currentAttackDelay <= 0)
            {
                //Updates player variable so animations can play appropriately
                player.SetAttacking(true);

                //Plays attacking sound effect
                SFXManager.PlaySound("PlayerAttack2");

                //Resets the attack delay
                currentAttackDelay = attackDelay;

                //MELEE ENEMIES
                //Creates a circle collider from the attack position and range and creates an array of any melee enemies inside the collider using a Layer Mask.
                Collider2D[] meleeEnemiesToDamage = Physics2D.OverlapCircleAll(attackPos.position, attackRange, whatIsMeleeEnemies);
                //Loops through the array above and makes any melee enemies take damage.
                for (int i = 0; i < meleeEnemiesToDamage.Length; i++)
                {
                    meleeEnemiesToDamage[i].GetComponent<EnemyMelee>().TakeDamage(damage);
                }

                //RANGED ENEMIES
                //Creates a circle collider from the attack position and range and creates an array of any ranged enemies inside the collider using a Layer Mask.
                Collider2D[] rangedEnemiesToDamage = Physics2D.OverlapCircleAll(attackPos.position, attackRange, whatIsRangedEnemies);
                //Loops through the array above and makes any ranged enemies take damage.
                for (int i = 0; i < rangedEnemiesToDamage.Length; i++)
                {
                    rangedEnemiesToDamage[i].GetComponent<EnemyRanged>().TakeDamage(damage);
                }

                //FLYING ENEMIES
                //Creates a circle collider from the attack position and range and creates an array of any flying enemies inside the collider using a Layer Mask.
                Collider2D[] flyingEnemiesToDamage = Physics2D.OverlapCircleAll(attackPos.position, attackRange, whatIsFlyingEnemies);
                //Loops through the array above and makes any flying enemies take damage.
                for (int i = 0; i < flyingEnemiesToDamage.Length; i++)
                {
                    flyingEnemiesToDamage[i].GetComponent<EnemyFlying>().TakeDamage(damage);
                }
            }
        }
        else
        {
            //Stops the player animation from showing the attacking varient
            player.SetAttacking(false);
        }
    }
}
```

Player Prefab



Collision Area



One collider array and for loop is required for each of the three enemy variants as they are each separate game object types. I would have liked to have simplified this to fit under one collider array and for loop however this caused too many issues and I moved onto other areas of the game. A YouTube tutorial was used to calculate this collider *(Blackthornprod, 2018)*.

The player attack also has an attack delay so the player cannot rapid click and kill enemies too quickly.

The YouTube clip below showcases the attack animation and the player attacking and killing one of the melee enemies.

Showcase (0:26): https://youtu.be/lzxFqtqxteU?t=26

Player Attack Script: https://pastebin.com/8RWjZTjs

Player (Take Damage)

Due to the isometric viewpoint I could not use the same hitbox as the one used for checking movement collisions. So I decided to keep the player damage script separate from the player controller so that it could use a different hitbox and not cause any issues.

The bulk of the code is simply a trigger enter collision check that looks for any collisions with the Damage tag. It also checks if the player can currently take damage.

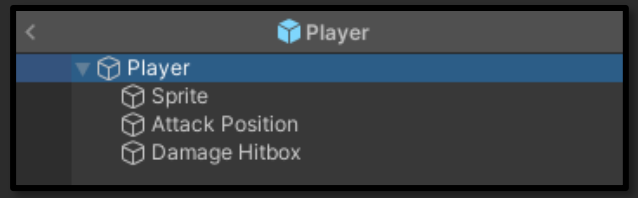
Then it makes the player take damage by running the take damage function on the player controller. This function checks if the players damage delay has reached 0 before lowering the players health. The hearts UI is updated at the same time to show the new health values.

This function also checks if the player has died. If the players health reaches 0 the death animation will play which will play the game over screen after the animation plays via an animation event.

Collision Detection (TakeDamage)

```
//Check if the player has collided with any trigger collisions
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //Checks if the player can currently take damage
    if (player.GetCanDamage())
    {
        //If the player collides with anything with the damage tag th
        if (collision.transform.CompareTag("Damage"))
        {
            player.TakeDamage(1);
        }
    }
}
```

Player Prefab



TakeDamage (PlayerController)

```
//---------- TAKE DAMAGE ----------
//Makes the player lose health, can be called from other scripts.
2 references
public void TakeDamage(int damage)
{
    //Checks if the player can take damage
    if (currentDamageDelay <= 0)
    {
        //Resets the damage delay
        currentDamageDelay = damageDelay;

        //Lower player health
        hearts -= damage;

        //Update hearts UI to show new values
        heartSystem.DrawHearts(hearts, maxHearts);

        //Triggers the colour change to show they have been damaged
        isDamaged = true;
        colourDelay = 0.25f;
    }

    //Checks if the player has died after taking damage
    if (hearts <= 0)
    {
        //Plays the appropriate death animation
        playerAnimator.Play("Death");

        //Sets the delay to allow the animation to play
        deathDelay = 2f;

        //Sets the bool to stop enemies from activating.
        dead = true;
    }
}
```

Player Controller Script: https://pastebin.com/KqrCVG71

Player Take Damage Script: https://pastebin.com/MZAJHHLe

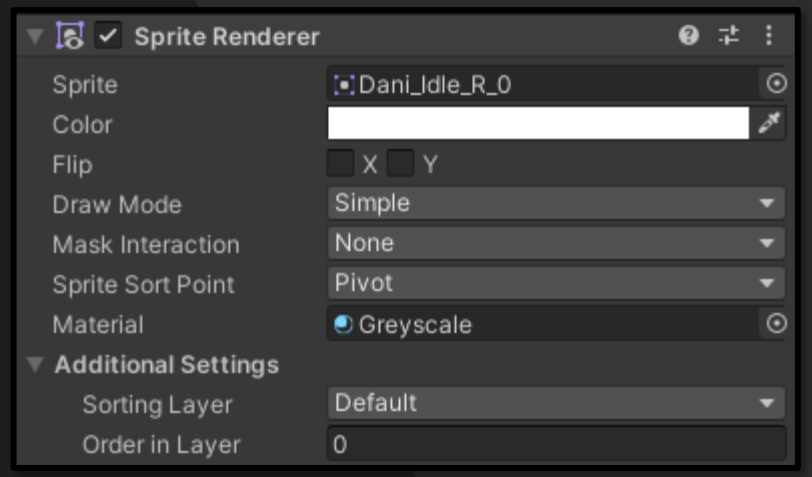
Showcase (0:34): https://youtu.be/lzxFqtqxteU?t=34

Player (Change Colour)

A script was created that would change the player sprite to greyscale when they take damage. This script checks if the player has been damaged and if so temporarily changes the sprites material to a greyscale shader.

This shader was taken from a YouTube tutorial *(Kee Gamedev, 2020)*

Greyscale Material applied to sprite



Updating colour dependant on player state

```
Unity Message | 0 references
private void Update()
{
    //If the player has been damaged the player sprite is set to greyscale.
    if (player.GetIsDamaged())
    {
        spriteRenderer.material.SetFloat("_GrayscaleAmount", 1);
    }
    //Otherwise there colour will be set to standard.
    else
    {
        spriteRenderer.material.SetFloat("_GrayscaleAmount", 0);
    }
}
```

Showcase (0:34): https://youtu.be/lzxFqtqxteU?t=34

Greyscale Shader: https://pastebin.com/8uVVAxLa

Player Colour Script: https://pastebin.com/yxerSG6e

Player (Animations)

I originally created player animations for idle, walking and attacking with placeholder assets while artwork was being developed. We had planned for player animations in 4 directions with a possibility of 8 directions if we had extra time. With this in mind I created a blend tree for 8 directions that could be easily adjusted to fit 4 directions later.

Unfortunately we were only able to get two directions for the player animations within the time constraints. This said I was still able to adjust the blend tree to four directions and reused the directions each twice to create four direction animations.

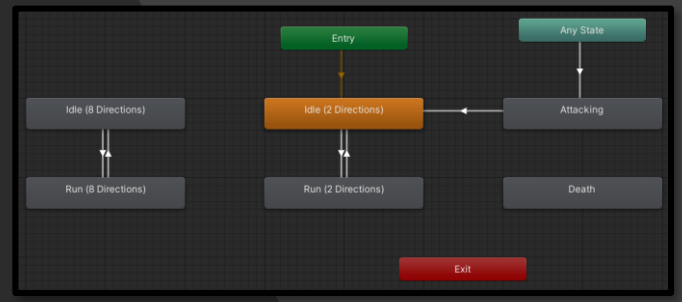
The player swaps between idle and walking automatically depending on the players magnitude value. Attack and death animations are played manually using animator.play() at the appropriate time.

Idle: https://youtu.be/lzxFqtqxteU?t=3
Walking: https://youtu.be/lzxFqtqxteU?t=7
Attack: https://youtu.be/lzxFqtqxteU?t=26
Death: https://youtu.be/lzxFqtqxteU?t=34

Death Animation Trigger

```
//Checks if the player has died after taking damage
if (hearts <= 0)
{
    //Plays the appropriate death animation
    playerAnimator.Play("Death");

    //Sets the delay to allow the animation to play
    deathDelay = 2f;

    //Sets the bool to stop enemies from activating.
    dead = true;
}
```

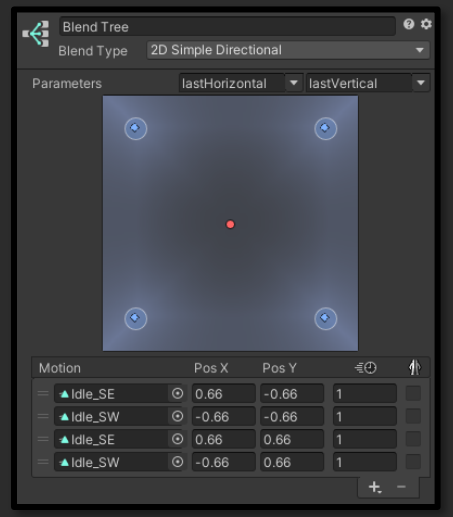
Animation Setup



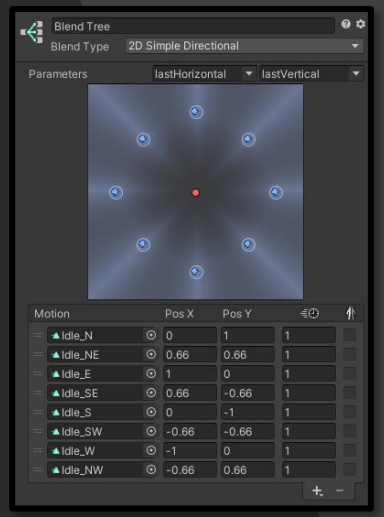
Updating Animations in PlayerController

```
//Updates the player animations so the correct animation shows at the appropriate time.
1 reference
private void UpdateAnimations()
{
    //Update movement variables needed for various animations
    playerAnimator.SetFloat("Horizontal", moveDirection.x);
    playerAnimator.SetFloat("Vertical", moveDirection.y);
    playerAnimator.SetFloat("Magnitude", moveDirection.magnitude);

    //Update attacking variable for the attack animation.
    playerAnimator.SetBool("Attacking", isAttacking);

    //Sets last direction variables for animations to allow them to play the correct one next (Idle, attacking)
    if (Input.GetAxisRaw("Horizontal") == 1 || Input.GetAxisRaw("Horizontal") == -1 || Input.GetAxisRaw("Vertical") == 1 || Input.GetAxisRaw("Vertical") == -1)
    {
        playerAnimator.SetFloat("lastHorizontal", Input.GetAxisRaw("Horizontal"));
        playerAnimator.SetFloat("lastVertical", Input.GetAxisRaw("Vertical"));
    }

    //Updates colour animation variables to ensure the correct colour bar varient is shown on each level.
    colourBorderAnimator.SetInteger("Level", currentLevel);
    colourFillAnimator.SetInteger("Level", currentLevel);

    //Update attack direction
    playerAnimator.SetFloat("MouseHorizontal", mouseWorldPos.x - transform.position.x);
    playerAnimator.SetFloat("MouseVertical", mouseWorldPos.y - transform.position.y);
}
```

Blend Tree (4 Directions)



Blend Tree (8 Directions)



Player Controller Script: https://pastebin.com/KqrCVG71

Sprite Render Order

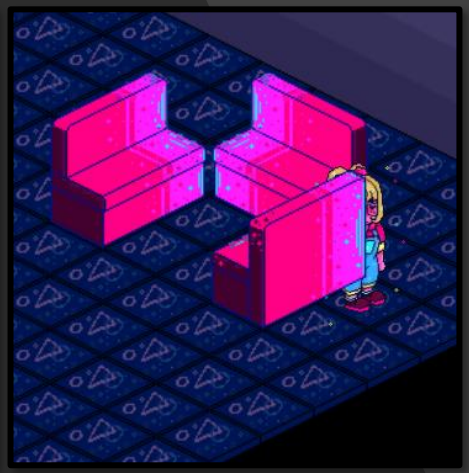
The sprite render order script was used to put the player in front and behind of objects in the scene for the isometric viewpoint. This script works by changing the sorting order of the player sprite automatically depending on their location in the scene. I used a YouTube tutorial *(Chris' Tutorials, 2018)* as a baseline while creating this script.

For the most part this has worked well with the player moving around objects as expected. However it is not perfect and there is some minor clipping issues when the player gets close to the edge of objects. With more time I would have liked to have figured out a better method for avoiding this clipping.

```
public class IsometricSpriteRenderer : MonoBehaviour
{
    //---------- DECLARE VARIABLES ----------
    public float value;
    private SpriteRenderer spriteRenderer;

    Unity Message | 0 references
    private void Start()
    {
        //Find sprite renderer
        spriteRenderer = GetComponent<SpriteRenderer>();
    }

    Unity Message | 0 references
    private void Update()
    {
        //Set the sprite sorting order relative to the players current position in the world.
        //Places them in front or behind of objects as they move around.
        spriteRenderer.sortingOrder = Mathf.RoundToInt(transform.position.y * value);
    }
}
```

Arcade Machines

Couches/Booth Seating



Sprite Render Order Script: https://pastebin.com/S6EAPFYu

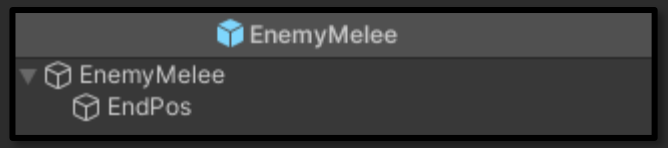
Melee Pigment (Movement)

The melee pigment is the basic pigment variant which moves between two points. A Boolean is used to see which way the enemy needs to move which toggles when the enemy reaches either end. The start position is stored when the pigment is spawned and the end position can be moved in engine via the EndPos transform in the prefab.

The pigment also features a rolling animation for moving. I controlled these manually in script as there were issues with the isometric format struggling to calculate which direction the enemy was facing.

EndPos Transform

Melee Enemy Prefab



Move to end position

```
//Checks if the enemy is set to move to the end.
if (moveToEnd == true)
{
    //Animations are updated to face the end position
    if (updateAnim)
    {
        //Float updates
        endX = endPos.x - transform.position.x;
        endY = endPos.y - transform.position.y;

        //Magnitude animation value updated
        animator.SetFloat("Magnitude", (endPos - transform.position).magnitude);

        //Calculates which direction the animation should face depending on the endX and endY floats.
        //End X calculation
        if (endX < 0)
        {
            animator.SetFloat("Horizontal", -1f);
        }
        else if (endX > 0)
        {
            animator.SetFloat("Horizontal", 1f);
        }
        //End Y calculation
        if (endY < 0)
        {
            animator.SetFloat("Vertical", -1f);
        }
        else if (endY > 0)
        {
            animator.SetFloat("Vertical", 1f);
        }

        //Sets the last horizontal and vertical directions for the death and attack animations.
        if (endPos.x - transform.position.x == 1 || endPos.x - transform.position.x == -1 || endPos.y - transform.position.y == -1 || endPos.y - transform.position.y == -1)
        {
            animator.SetFloat("Last Horizontal", (endPos.x - transform.position.x));
            animator.SetFloat("Last Vertical", (endPos.y - transform.position.y));
        }

        //Flips the update animation boolean so it can update to face the other direction.
        updateAnim = false;
    }

    //Moves the enemy towards the end position at "step" speed
    transform.position = Vector3.MoveTowards(transform.position, endPos, step);
}
```

Move back to start

```
else
{
    //Enemy animations are updated to face start position
    if (updateAnim == false)
    {
        //Magnitude animation value updated
        animator.SetFloat("Magnitude", (startPos - transform.position).magnitude);

        //Float updates
        startX = startPos.x - transform.position.x;
        startY = startPos.y - transform.position.y;

        //Calculates which direction the animation should face depending on the direction of startX and startY.
        //Start X Calculations
        if (startX < 0)
        {
            animator.SetFloat("Horizontal", -1f);
        }
        else if (startX > 0)
        {
            animator.SetFloat("Horizontal", 1f);
        }
        //Start Y calculations
        if (startY < 0)
        {
            animator.SetFloat("Vertical", -1f);
        }
        else if (startY > 0)
        {
            animator.SetFloat("Vertical", 1f);
        }

        //Sets the last horizontal and vertical directions for the death and attack animations.
        if (startPos.x - transform.position.x == 1 || startPos.x - transform.position.x == -1 || startPos.y - transform.position.y == -1 || startPos.y - transform.position.y == -1)
        {
            animator.SetFloat("Last Horizontal", (startPos.x - transform.position.x));
            animator.SetFloat("Last Vertical", (startPos.y - transform.position.y));
        }
        //Flips update animation boolean so it can update animations facing the other direction
        updateAnim = true;
    }

    //Moves the enemy towards the start position at "step" speed.
    transform.position = Vector3.MoveTowards(transform.position, startPos, step);
}
```

Showcase (0:55): https://youtu.be/lzxFqtqxteU?t=55

Melee Pigment Script: https://pastebin.com/TsHGr1wd

Melee Pigment (Attack/Hurt/Death)

The enemy attack is primarily controlled in the player controller as shown earlier as the pigment is given the damage tag in engine.

The animations however are controlled in the enemy script with an if statement checking which level the enemy is on.

```
if (collision.transform.CompareTag("Player Hitbox"))
{
    //Pink Attack (Level 1)
    if (currentLevel == 1)
    {
        animator.Play("Pink Attack");
    }
    //Cyan Attack (Level 2)
    else if (currentLevel == 2)
    {
        animator.Play("Cyan Attack");
    }
    //Yellow Attack (Level 3)
    else
    {
        animator.Play("Yellow Attack");
    }
```

The enemy will take damage when the TakeDamage function is called from the player attack script.

This will first check if the enemy is still alive then stun the enemy for a set amount of time before lowering their health.

This function also plays the hurt animations by checking which level the enemy is on and playing the appropriate animation.

```
public void TakeDamage(int damage)
{
    //If the enemys health is above 0 the fol
    if (health > 0)
    {
        //Hurt sound effect
        SFXManager.PlaySound("RollieHurt");
        //The stun delay is reset
        currentStunDelay = stunDelay;

        //The enemy takes damage
        health -= damage;
    }

    //If the enemy has not died the appropria
    if (dead == false)
    {
        //Pink Hurt (Level 1)
        if (currentLevel == 1)
        {
            animator.Play("Pink Hurt");
        }
        //Cyan Hurt (Level 2)
        else if (currentLevel == 2)
        {
            animator.Play("Cyan Hurt");
        }
        //Yellow Hurt (Level 3)
        else if (currentLevel == 3)
        {
            animator.Play("Yellow Hurt");
        }
    }
}
```

The destroy pigment function is run at the end of the death animation via an animation event.

This adds colour to the colour bar and then destroys the game object.

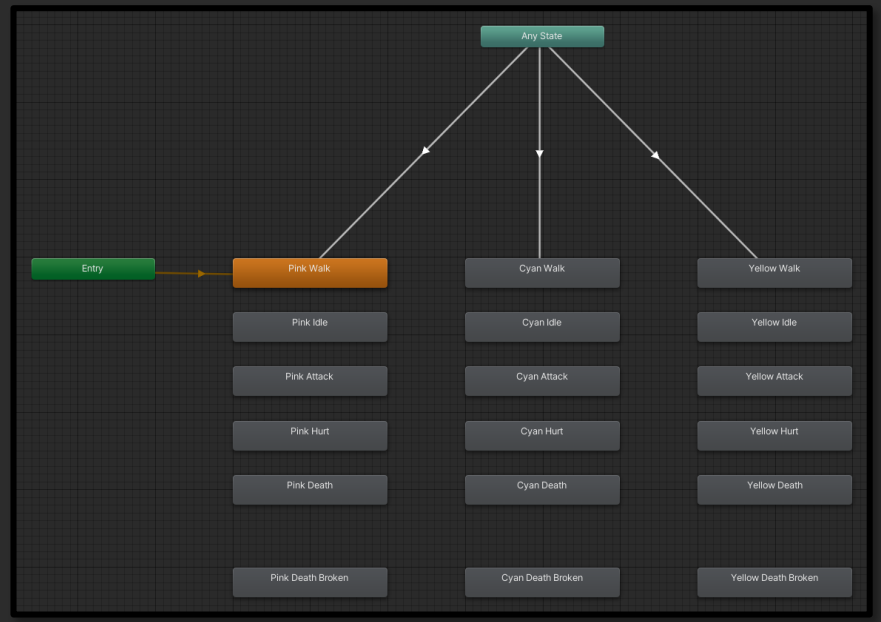
```
public void DestroyPigment()
{
    player.AddColour(colourValue);
    Destroy(gameObject);
}
```

The check health function is run constantly, checking if the enemy has died by updating when the enemy's health reaches 0.

This will destroy the hitbox of the enemy to stop it damaging the player and then play the death animation of the pigment,

```
private void CheckHealth()
{
    //If the enemy reaches 0 health some bool
    if (health <= 0)
    {
        dead = true;
        move = false;
    }

    //Once the enemy dies this will run
    if (dead)
    {
        //Destroys the hitbox so they cannot
        Destroy(hitbox);

        //Play Death Animation relative to cu
        //Dead boolean is set to false to sto
        //Pink Varient (Level 1)
        if (currentLevel == 1)
        {

            animator.Play("Pink Death");
            dead = false;
        }
        //Cyan Varient (Level 2)
        else if (currentLevel == 2)
        {
            animator.Play("Cyan Death");
            dead = false;
        }
        //Yellow Varient (Level 3)
        else
        {
            animator.Play("Yellow Death");
            dead = false;
        }
    }
}
```

Animation Setup



Showcase (0:55): https://youtu.be/lzxFqtqxteU?t=55

Melee Pigment Script: https://pastebin.com/TsHGr1wd

Ranged Pigment

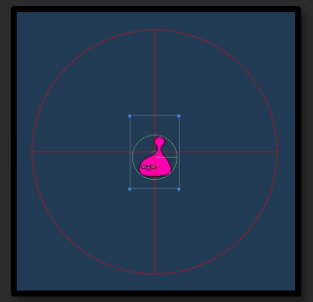
The ranged enemy is a stationary variant that activates when the player gets in range. Once in range, the enemy will continually shoot projectiles at the player.

The ranged pigment was originally scripted by Harry, however after working on the Lizard Wizard boss myself I reworked the ranged pigment with a simpler method.

The rework features if statements to figure out if the enemy is allowed to fire. If the enemy can fire, an attack animation is played and a projectile is spawned on top of the enemy.

As with the melee pigment, this enemy takes damage and is destroyed with the same method.

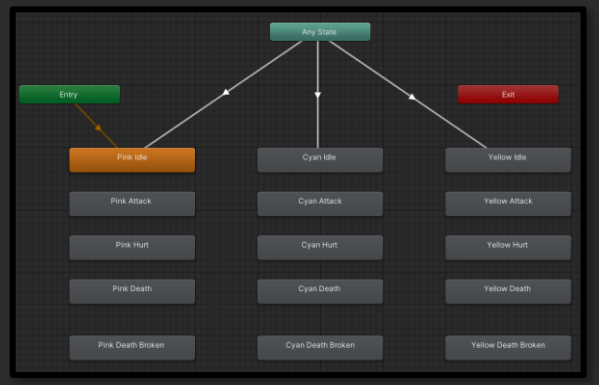
To check if the player is in range a circle collider is created around the enemy with a modifiable range that checks if the player has collided with it.



Circle collider / Is player in range

```
//Creates a circle collider that looks for the player
Collider2D inRange = Physics2D.OverlapCircle(transform.position, attackRange, whatIsPlayer);

//If the player is in the collider range the enemy can start shooting
if (inRange)
{
    canShoot = true;
}
//Otherwise they will be unable to shoot.
else
{
    canShoot = false;
}
```

Animation Setup



Update Animations

As the ranged enemy is stationary, I decided to make the animations face the player instead.

```
private void UpdateAnimations()
{
    animator.SetFloat("Horizontal", player.transform.position.x - transform.position.x);
    animator.SetFloat("Vertical", player.transform.position.y - transform.position.y);
}
```

Can enemy fire? / Spawn Projectile

```
//---------- FIXED UPDATE ----------
Unity Message | 0 references
private void FixedUpdate()
{
    //If the player is alive
    if (player.GetIsDead() == false)
    {
        //If the players room number matches the enemys room number
        if (roomNumber == player.GetRoomNumber())
        {
            //Enemy is set to active
            isActive = true;
        }
        //While the enemy is alive
        if (dead == false)
        {
            //If enemy is active
            if (isActive)
            {
                //Cant fire if the enemy has just been damaged
                if (hurt == false)
                {
                    //If the enemy can shoot
                    if (canShoot)
                    {
                        //if the fire delay is low enough
                        if (Time.time > nextFire)
                        {
                            //play attack sound effect
                            SFXManager.PlaySound("ThrowieAttack");
                            //Plays attack animation depending on what level it is.
                            //Pink (Level 1)
                            if (currentLevel == 1)
                            {
                                animator.Play("Pink Attack");
                            }
                            //Cyan (Level 2)
                            else if (currentLevel == 2)
                            {
                                animator.Play("Cyan Attack");
                            }
                            //Yellow (Level 3)
                            else if (currentLevel == 3)
                            {
                                animator.Play("Yellow Attack");
                            }

                            //Spawns a projectile on top of the enemy
                            Instantiate(projectile, transform.position, Quaternion.identity);

                            //Reset the fire rate counter
                            nextFire = Time.time + fireRate;
                        }
                    }
                }
            }
        }
    }
}
```

Showcase (1:19): https://youtu.be/lzxFqtqxteU?t=79

Ranged Enemy Script: https://pastebin.com/i79k9MQY

Ranged Pigment Projectile

The ranged pigments projectile functions by locating the player and moving the rigidbody towards them.

The projectile is constantly checking for collisions with either the player or tilemap collisions to know if it should be destroyed.

There is also a timer that destroys the game object after a period of time if it has not hit anything.

While the projectiles are static images, I used animations to update the sprite with the appropriate variant for the level.

A YouTube tutorial helped with calculating the players position and moving the projectile towards them. *(Zotov, A., 2018)*

Initialise Projectile

```
//---------- SET UP PROJECTILE AND FIRE AT PLAYER ----------
Unity Message | 0 references
private void Start()
{
    //Projectile SFX
    SFXManager.PlaySound("ThrowieAttack");
    //Find Game objects
    player = GameObject.FindObjectOfType<PlayerController>();
    animator = GetComponent<Animator>();
    rb = GetComponent<Rigidbody2D>();

    //set current level to the same as the players
    currentLevel = player.GetCurrentLevel();

    //Set animations relative to the current level
    animator.SetInteger("Current Level", currentLevel);

    //Set move direction towards the player
    moveDirection = (player.transform.position - transform.position).normalized * moveSpeed;

    //Move the projectile towards the player
    rb.velocity = new Vector2(moveDirection.x, moveDirection.y);

    //Destroy the projectile after 3f
    Destroy(gameObject, 3f);
}
```

Collision Detection

```
//---------- CHECK FOR COLLISIONS -----------
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //Check if the projectile has collided with the player hitbox
    if (collision.transform.CompareTag("Player Hitbox"))
    {
        Destroy(gameObject);
    }

    //Check if the projectile has collided with any collisions in
    if (collision.transform.CompareTag("Collisions"))
    {
        Destroy(gameObject);
    }
}
```

Showcase (1:19): https://youtu.be/lzxFqtqxteU?t=79

Ranged Enemy Projectile Script: https://pastebin.com/XCSjGSuL

Flying Pigment

The flying pigment starts inactive and when the player moves into range they will activate and follow the player attacking them. If the player gets out of range they will return to the starting position.

While Harry handled the movement of the flying enemy including the player tracking and range activation, I added the extra functions to the enemy such as activating per room, taking damage, destroying the enemy after death and collision detection.

I also implemented the enemy animations into engine. These animations updated to face the player similar to the ranged enemy before. However when they return to their starting position after moving out of range they will face the starting position instead.

Animations towards start

```
//Updates animations relative to the start position
animator.SetFloat("Horizontal", (startPos.x - transform.position.x));
animator.SetFloat("Vertical", (startPos.y - transform.position.y));
```

Animations towards player

```
//Animations
animator.SetFloat("Horizontal", (player.transform.position.x - transform.position.x));
animator.SetFloat("Vertical", (player.transform.position.y - transform.position.y));
```

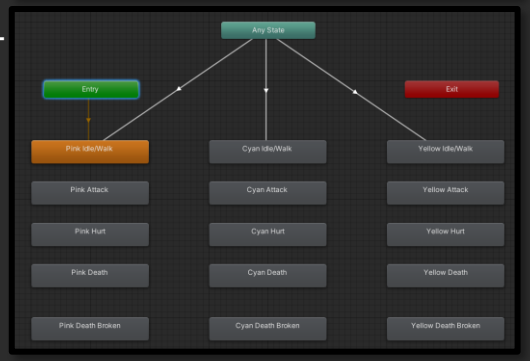
Check enemy health

```
//---------- CHECK IF ENEMY HAS DIED ----------
1 reference
private void CheckHealth()
{
    //If the enemy reaches 0 health they are set to dead
    if (health <= 0)
    {
        dead = true;
    }

    //Once the enemy dies this will run
    if (dead)
    {
        //Destroys the hitbox so they cannot hurt the player anymore
        Destroy(hitbox);

        //Play Death Animation relative to current level
        //Pink Varient (Level 1)
        if (currentLevel == 1)
        {
            animator.Play("Pink Death");
            dead = false;
        }
        //Cyan Varient (Level 2)
        else if (currentLevel == 2)
        {
            animator.Play("Cyan Death");
            dead = false;
        }
        //Yellow Varient (Level 3)
        else
        {
            animator.Play("Yellow Death");
            dead = false;
        }
    }
}
```

Collision Detection

```
//---------- CHECK IF ENEMY HAS COLLIDED WITH PLAYER -----
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //If the enemy collides with the player hitbox the att
    if (collision.transform.CompareTag("Player Hitbox"))
    {
        //attacking sound effect
        SFXManager.PlaySound("FlyingAttack");
        //Pink Attack (Level 1)
        if (currentLevel == 1)
        {
            animator.Play("Pink Attack");
        }
        //Cyan Attack (Level 2)
        else if (currentLevel == 2)
        {
            animator.Play("Cyan Attack");
        }
        //Yellow Attack (Level 3)
        else
        {
            animator.Play("Yellow Attack");
        }
    }
}
```

Animation Setup



Destroy Pigment

```
public void DestroyPigment()
{
    player.AddColour(colourValue);
    Destroy(gameObject);
}
```

Take Damage

```
//---------- ENEMY TAKES DAMAGE ----------
1 reference
public void TakeDamage(int damage)
{
    //If the enemys health is above 0 the following code will run
    if (health > 0)
    {
        //hurt sound effect
        SFXManager.PlaySound("FlyingHurt");
        //The enemy takes damage
        health -= damage;
    }

    //If the enemy has not died the appropriate hurt animations will be played
    if (dead == false)
    {
        //Pink Varient (Level 1)
        if (currentLevel == 1)
        {
            animator.Play("Pink Hurt");
        }
        //Cyan Varient (Level 2)
        else if (currentLevel == 2)
        {
            animator.Play("Cyan Hurt");
        }
        //Yellow Varient (Level 3)
        else
        {
            animator.Play("Yellow Hurt");
        }
    }
}
```

Showcase (1:45): https://youtu.be/lzxFqtqxteU?t=105

Flying Pigment Script: https://pastebin.com/cgjhHLrw

Lizard Wizard

The Lizard Wizard is the final boss of the game and has three attack stages: Shockwave, Pigments and Ranged. These stages are managed using a state machine. Alongside this many other functions were created to create the boss as it is in the final game.

Take Damage

The Take Damage function is similar to previous enemies and lowers the lizards health and plays an animation. It also toggles a boolean which temporarily stops the lizard from attacking until the hurt animation finishes. The healthbar is also updated here.

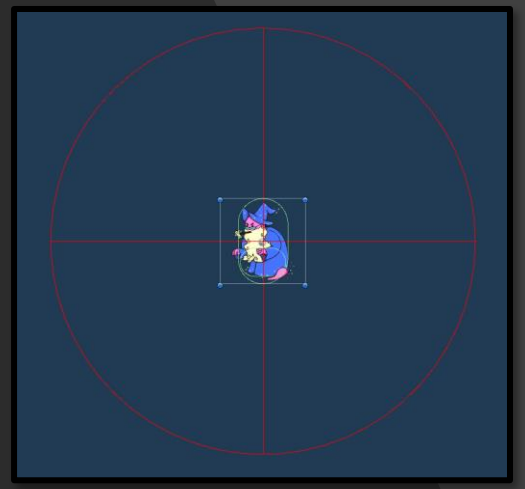
```
//---------- LIZARD WIZARD TAKES DAMAGE ----------
3 references
public void TakeDamage(int damage)
{
    //Plays a damage effect
    SFXManager.PlaySound("BossHurt");

    animator.Play("Hurt");
    hurt = true;

    //Lowers the lizard wizards health
    currentHealth -= damage;

    //Updates the healthbar
    UpdateHealth(currentHealth / maxHealth);
}
```

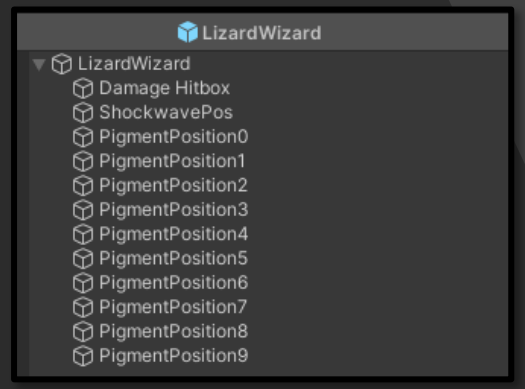
Check Dead

The check dead function updates booleans if the lizard wizard reaches 0 health which will trigger animations, sound effects and more which lead to the game progressing.

```
//---------- CHECK IF LIZARD HAS DIED ----------
1 reference
private void CheckDead()
{
    if (isActive)
    {
        //If the lizard has reached 0 health the
        if (currentHealth <= 0)
        {
            //Death sound effect plays
            SFXManager.PlaySound("BossDeath1");

            //Boolean toggle
            destroyed = true;

            //Destroy the game object
            //Destroy(gameObject);

            isActive = false;
        }
    }
}
```

Lizard Wizard Prefab





Showcase (3:49): https://youtu.be/lzxFqtqxteU?t=229

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard (Animations)

The lizard wizard has many animations that play at various points in the boss fight. These are: Idle, Attack, Slam, Spawn, Hurt and Death. Different controls and triggers were needed to play these animations at the right time. I also used animation events to control some features of the lizard wizard such as fire rates for different stages and triggering the end of game condition in the death animation.

Update Sprites / Animations

Animations and sprite updates are done through the UpdateSprite function. An if statement checks which state the lizard is in and will hide the sprite renderer and disable their hitbox if in either of the pigment states. Animations are also updated to face the player at all times.

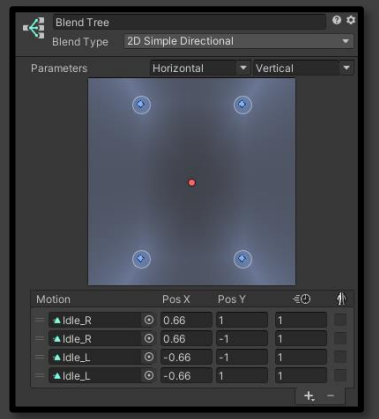
```
//---------- UPDATE SPRITE ----------
1 reference
private void UpdateSprite()
{
    //Depending on what state is active the lizard wizard sprite will be visible or invisible.
    //The hitbox will also be toggled depending on whether they are visible or not.
    if (state == State.Pigments || state == State.PigmentInactive)
    {
        spriteRenderer.enabled = false;
        hitbox.enabled = false;
    }
    else
    {
        spriteRenderer.enabled = true;
        hitbox.enabled = true;
    }

    //Animation Updates
    animator.SetFloat("Horizontal", player.transform.position.x - transform.position.x);
    animator.SetFloat("Vertical", player.transform.position.y - transform.position.y);
```

Idle: https://youtu.be/lzxFqtqxteU?t=136
Ranged: https://youtu.be/lzxFqtqxteU?t=179
Shockwave: https://youtu.be/lzxFqtqxteU?t=216
Spawn: https://youtu.be/lzxFqtqxteU?t=132
Hurt / Death: https://youtu.be/lzxFqtqxteU?t=229

As with the player, the lizard uses blend trees to swap between each of the two directions. I again had to use the animations twice since we did not have 4 directions.

Many animations are triggered using animator.play to play the animation once. Using animation events I was able to trigger certain functions such as death and hurt booleans that toggled the lizard wizards states.

Idle Blend Tree



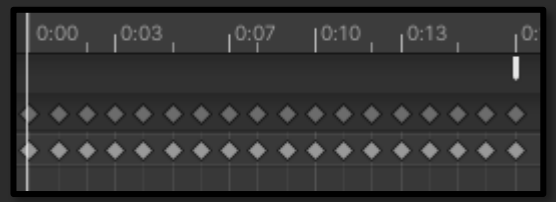
Trigger Animations

```
animator.Play("Spawn");
```

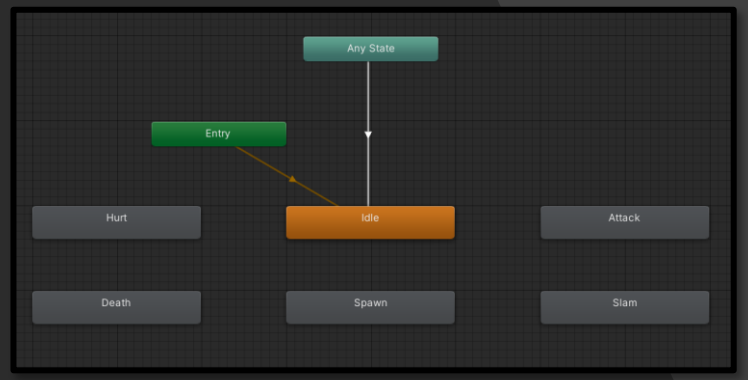
```
animator.Play("Hurt");
```

```
if (destroyed)
{
    state = State.Inactive;
    animator.Play("Death");
}
```

Hurt Animation Event



Animation Setup



Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard (Stages)

The LizardWizard is the final boss of the game and has three attack stages: Shockwave, Pigments and Ranged. A state machine holds the different states the boss uses. These states are managed in a Choose Stage function that updates constantly.

First the function checks if there is any delay currently active and if there is it will count down using Time.deltaTime.

It then checks if the boss is active and uses an if statement to find the current stage, delay or other values to decide which stage to pick next.

Each if statement will change the state to a different one depending on which current stage is active. The order and change conditions are:

1: Shockwave: Is active for a set period of time before moving to the next stage.

2: Pigments: Active until five pigments have spawned then moves to the next stage.

3: Pigments Inactive: Waits until all five pigments have been killed before moving to the next stage.

4: Ranged: Is active for a set period of time before moving to the next stage.

Duration / Shockwave -> Pigment

```
private void ChooseStage()
{
    //Counts down the stage duration as long as it is above 0
    if (currentStageDuration > 0)
    {
        currentStageDuration -= Time.deltaTime;
    }

    //If the boss is active it will move between stages
    if (isActive)
    {
        //Move from shockwave to pigments
        if (currentStage == 2 && currentStageDuration <= 0)
        {
            //Reset pigment amount
            pigmentAmount = 0;

            //Reset pigment list to get fresh positions
            SetPigmentList();

            //Set new stage
            currentStage = 3;
            state = State.Pigments;
        }
```

Pigments -> Ranged

```
        //move from pigments to ranged
        if (currentStage == 3 && pigmentStageEnd == true)
        {
            //Reset boolean
            pigmentStageEnd = false;

            //Boss takes damage as all pigments have died
            TakeDamage(3);

            //Sets stage duration to ranged duration
            currentStageDuration = maxRangedDuration;

            //Sets new stage
            currentStage = 4;
            state = State.Ranged;
        }
```

Pigments -> Pigments Inactive

```
        //Pause pigment stage when 5 pigments are on screen
        else if (currentStage == 3 && pigmentAmount >= 5)
        {
            //Change state
            state = State.PigmentInactive;

            //Reset pigment amount
            pigmentAmount = 0;
        }
```

Ranged -> Shockwave

```
        //move from ranged to shockwave
        if (currentStage == 4 && currentStageDuration <= 0)
        {
            //Sets stage duration to shockwave duration
            currentStageDuration = maxShockwaveDuration;

            //Sets new stage
            currentStage = 2;
            state = State.Shockwave;
        }
    }
}
```

Showcase (2:20): https://youtu.be/lzxFqtqxteU?t=140

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard (Ranged)

The lizards ranged stage fires projectiles at the player repeatedly. These projectiles can be dodged or hit back at the lizard to deal damage.

An animation event on the lizard wizard attack animation controls the fire rate of this stage so projectiles spawn at the appropriate time.

Update if lizard can fire

Check if lizard can fire a projectile

```
case State.Ranged:

    //Play animation
    //The attack animation controls the firerate of the projectiles by toggling
    animator.Play("Attack");

    //Checks if the lizard can fire. (Can't fire if they are taking damage)
    if (rangedFire && hurt == false)
    {
        //Play sound effect
        SFXManager.PlaySound("BossProjectile");

        //Spawn projectile on the lizard wizard
        Instantiate(rangedProjectile, transform.position, Quaternion.identity);

        //Toggle the fire boolean off.
        rangedFire = false;
    }
```

The ranged projectiles spawn when the animation has toggled the ranged fire boolean and if they are not taking damage.

```
0 references
public void ToggleRangedFire()
{
    rangedFire = true;
}
0 references
```

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Showcase (2:59): https://youtu.be/lzxFqtqxteU?t=179

Lizard Wizard Ranged Script: https://pastebin.com/PiZg053V

Lizard Wizard (Ranged)

The fired projectile will default to target the player and uses a state machine to toggle between moving to the player and moving back to the lizard.

To check when the player is attacking a collider is made around the projectile that checks if the player is in range and attacking. If so the state will be swapped to the to lizard state.

Collision Detection

```
//---------- COLLISION DETECTION ----------
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //Checks if the projectile has collided with the player a
    if (collision.transform.CompareTag("Player Hitbox"))
    {
        Destroy(gameObject);
    }

    //If the projectile can damage the lizard and collides wi
    if (canDamageLizard)
    {
        if (collision.transform.CompareTag("Lizard Hitbox"))
        {
            //Sound effect plays
            SFXManager.PlaySound("Boss");
            lizardWizard.TakeDamage(damage);

            //Projectile is destroyed
            Destroy(gameObject);
        }
    }

    //If the projectile collides with any tilemap collisions
    if (collision.transform.CompareTag("Collisions"))
    {
        Destroy(gameObject);
    }
}
```

The projectile should not damage the lizard while moving towards the player therefore I made a boolean toggle that enables and disables the collision check. This boolean is changed in the state machine.

If the projectile collides with the player or collisions it will be destroyed.

If the lizard is hit by the projectile when it returns it will take damage via the TakeDamage function in the lizard script.

Set up default projectile values

```
//Set move direction
moveDirection = (player.transform.position - transform.position).normalized * moveSpeed;

//Destroy gameobject after 5f
Destroy(gameObject, 5f);

//Set default state as to player
state = State.ToPlayer;
```

Swap state if player is in range and attacking

```
//Create a collider and check if the player is inside the collider
Collider2D inRange = Physics2D.OverlapCircle(returnPos.position, returnRange, whatIsPlayer);

//Checks if the player is in range
if (inRange)
{
    //Checks if the player is attacking and starts returning the projectile to the lizard.
    if (player.GetAttacking())
    {
        state = State.ToLizardWizard;
    }
}
```

State machine

```
//State machine
switch (state)
{
    //Moves the projectile to the player
    case State.ToPlayer:
        rb.velocity = new Vector2(moveDirection.x, moveDirection.y);
        //Stops the projectile damaging the lizard wizard
        canDamageLizard = false;

        break;


    //Moves the projectile to the Lizard Wizard
    case State.ToLizardWizard:
        //Lets the projectile damage the lizard wizard
        canDamageLizard = true;

        //Calculates a new move direction and moves it towards the lizard wizard
        moveDirection = (lizardWizard.transform.position - transform.position).normalized * moveSpeed;
        rb.velocity = new Vector2(moveDirection.x, moveDirection.y);
        break;
}
```

Showcase (2:59): https://youtu.be/lzxFqtqxteU?t=179

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard Ranged Script: https://pastebin.com/PiZg053V

Lizard Wizard (Pigments)

I designed the pigment stage which features five random pigments spawning in random locations while the boss disappears. The player has to defeat these pigments to deal large damage to the boss who will then reappear in the next attack state.

First I created a list to store all the pigment positions to be gathered later. Game objects were made in the lizard wizard prefab that could be placed around the boss level as spawn points. These are looped through storing each one in the list. This list is reset at the start of each pigment stage.

Set up list of pigment positions

```
private void SetPigmentList()
{
    //Clears the list so it can be recreated with new values
    list.Clear();

    //Loops through the pigment spawns adding their positions to the list
    for (int i = 0; i < pigmentSpawns.Length; i++)
    {
        pigmentSpawn = pigmentSpawns[i];

        //Inserts the new pigment spawn position into the list
        list.Insert(i, (new Vector3(pigmentSpawn.transform.position.x, pigmentSpawn.transform.position.y)));

        //converts the list index to string value for debugging
        listText = list[i].ToString();
    }
}
```

Get pigment type and random position

```
//Only five pigments will be spawned per stage
if (pigmentAmount < 5)
{
    //Controls the spawnrate of the pigments so they do not appear all at once
    if (Time.time > spawnNext)
    {
        //Stores the last pigment type
        lastPigment = pigmentType;
        //Constantly loops to ensure the same pigment does not spawn straight after itself
        {
            //Random range for the pigment types
            pigmentType = Random.Range(1, 4);
        }

        //Picks a random number between 0 and the amount of values in the list of positions
        int index = Random.Range(0, list.Count);
        //Stores the vector position of the list value.
        Vector3 randomPigmentPosition = list[index];
        //Removes the value from the list to ensure no duplicates
        list.RemoveAt(index);
```

The pigment stage will only spawn five enemies and will never spawn two of the same type in a row, using a while loop and random integer value.

A random position is also gathered from the list made earlier. The value chosen is removed from the list to ensure no pigments spawn in the same place.

Lizard Wizard Prefab

Showcase (3:09): https://youtu.be/lzxFqtqxteU?t=189

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard (Pigments)

After calculating which pigment type will spawn and the random spawn position, an if statement is used to spawn the correct pigment in the random position.

The pigment is set to active which lets them move around in the boss level as they have previously.

The spawn timer is reset and the pigment amount is increased after spawning each pigment.

Spawn correct pigment

```
//Controls how the pigment spawns
if (pigmentType == 1)
{
    //Plays a pigment spawn sound effect
    //SFXManager.PlaySound("BossProjectile2");

    //Spawns a melee enemy at the random position calculated earlier
    GameObject pigment = Instantiate(enemyMelee, randomPigmentPosition, Quaternion.identity);

    //Gets the melee pigment so it can be activated.
    EnemyMelee thePigment = pigment.GetComponent<EnemyMelee>();

    //Activates the melee pigment
    thePigment.SetActive(true);

}
//Ranged Pigment
else if (pigmentType == 2)
{
    //Plays a pigment spawn sound effect
    //SFXManager.PlaySound("BossProjectile2");

    //Spawns a ranged enemy at the random position calculated earlier
    GameObject pigment = Instantiate(enemyRanged, randomPigmentPosition, Quaternion.identity);

    //Gets the ranged pigment so it can be activated.
    EnemyRanged thePigment = pigment.GetComponent<EnemyRanged>();

    //Activates the ranged pigment
    thePigment.SetActive(true);
}
//Flying Pigment
else if (pigmentType == 3)
{
    //Plays a pigment spawn sound effect
    //SFXManager.PlaySound("BossProjectile2");

    //Spawns a flying enemy at the random position calculated earlier
    GameObject pigment = Instantiate(enemyFlying, randomPigmentPosition, Quaternion.identity);

    //Gets the flying pigment so it can be activated.
    EnemyFlying thePigment = pigment.GetComponent<EnemyFlying>();

    //Activates the flying pigment
    thePigment.SetActive(true);
}

//Resets the spawn rate
spawnNext = Time.time + spawnRate;

//Adds to the pigment amount so no more than 5 can spawn
pigmentAmount += 1;
```

Inactive Pigment State

The inactive pigment state is used when five pigments have been spawned.

This searches for any pigments in the scene and if there are none left will end the pigment stage and move to the next stage.

```
//The inactive state is used to pause the boss while the player kills
case State.PigmentInactive:

    //Checks if there are any pigments in the scene
    if (GameObject.FindGameObjectWithTag("Damage") != null)
    {
        //If there are pigments present the stage will not end
        pigmentStageEnd = false;
    }
    //If there are no pigments left the stage will end and move on.
    else
    {
        pigmentStageEnd = true;
    }
    break;
```

Showcase (3:09): https://youtu.be/lzxFqtqxteU?t=189

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard (Shockwave)

The shockwave was designed to be a quick burst attack to take the player by surprise with the stage not lasting very long. This attack uses 16 projectiles that fire out in a circle from the lizard wizards position damaging the player if hit. If the projectile hits either the player or the tilemap collision it will be destroyed.

The projectiles are spawned in the lizard wizard script and controlled via the Lizard Wizard Shockwave script.

I first created an array of vectors to store and access the different directions each shockwave projectile would fire in.

A collider2D is used to check if the player is within shockwave range before it begins firing. The fire rate is controlled by the animation using an animation event that toggles a boolean to ensure it syncs up correctly.

When the lizard is allowed to start firing, the script uses a for loop to go through the direction array, spawning a new projectile for each one and setting its move direction to the vector.

Finally the boolean that controls the fire rate is set to false to stop the lizard firing.

Move projectile to vector direction

```
private void FixedUpdate()
{
    //Moves the projectile in the move direction set in the lizard wizard script
    rb.velocity = new Vector2(moveDirection.x, moveDirection.y).normalized * moveSpeed; ;
}
```

Toggle fire

```
public void ToggleShockwaveFire()
{
    shockwaveFire = true;
}
```

Destroy projectile on collision

```
private void OnTriggerEnter2D(Collider2D collision)
{
    //If the projectile collides with the player the projectile is destroyed
    if (collision.transform.CompareTag("Player Hitbox"))
    {
        Destroy(gameObject);
    }

    //If the projectile collides with the tilemap collisions it is destroyed
    if (collision.transform.CompareTag("Collisions"))
    {
        Destroy(gameObject);
    }
}
```

Spawn projectiles

```
//This state controls the shockwave attack
case State.Shockwave:

    //A collider is made to check if the player enters the range that the lizard can fire the shockwave from.
    Collider2D inRange = Physics2D.OverlapCircle(shockwavePos.position, shockwaveRange, whatIsPlayer);

    //If the player is in range the lizard will start firing shockwaves at the appropriate firerate
    if (inRange)
    {
        //Play animation
        animator.Play("Slam");

        //Waits for toggle from animation
        if (shockwaveFire)
        {
            //Loops through the shockwave direction array and spawns a projectile that fires once in each of the d
            for (int i = 0; i < shockwaveDirectionArray.Length; i++)
            {
                //Sets the shockwave direction from the array
                shockwaveDirection = shockwaveDirectionArray[i];
                //Spawns a shockwave projectile on the lizard wizard
                GameObject shockwave = Instantiate(shockwaveProjectile, transform.position, Quaternion.identity);
                //Gets the shockwave projectile component so its direction can be updated
                LizardWizardShockwave theShockwave = shockwave.GetComponent<LizardWizardShockwave>();
                //Updates the shockwave projectile direction so it fires in a unique direction.
                theShockwave.SetMoveDirection(shockwaveDirection);

            }
            //Plays the shockwave attack sound
            SFXManager.PlaySound("BossProjectile3");

            //Stops shockwaves firing
            shockwaveFire = false;
```

Showcase (3:36): https://youtu.be/lzxFqtqxteU?t=216

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Lizard Wizard Shockwave Script: https://pastebin.com/0ifVVYft

Healthpack

A healthpack script and prefab was made to restore player health while navigating the levels.

The script first checks if the player has lost health otherwise the player cannot pickup the Healthpack. Then it checks if the player is colliding with the object and will destroy it if so and then run the AddHealth() function in the PlayerController.

In the player controller the AddHealth() function first checks if the player has lost health and if so will add the amount inputted to the players current health. It will also update the hearts shown on the UI.

Finally to make the Healthpack work with each all levels in all colours I created an animation controller that updates the animation depending on what level is selected.

Collision detection

```
//---------- COLLISION DETECTION ----------
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //If the player has lost health they can collide w
    if (player.GetHealth() < player.GetMaxHealth())
    {
        if (collision.transform.CompareTag("Player"))
        {
            Destroy(gameObject);
            player.AddHealth(1);
        }
    }
}
```

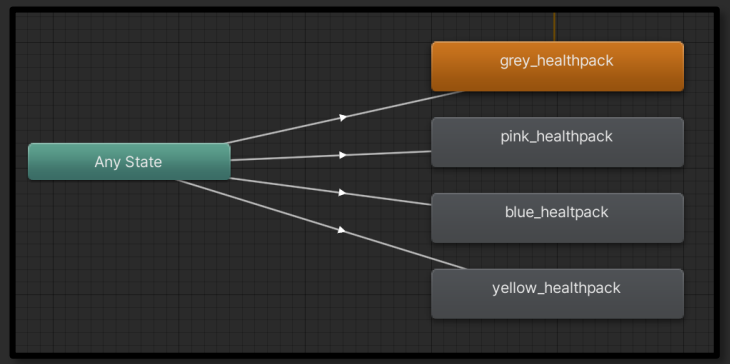
Adds to the player health

```
public void AddHealth(int amount)
{
    //Checks if the player is under max health and
    if (hearts < maxHearts)
    {
        hearts += amount;
        //Debug.Log("Current Health: " + hearts);

        //Updates UI with the new values
        heartSystem.DrawHearts(hearts, maxHearts);
    }
}
```

Update animations per level

```
Unity Message | 0 references
private void Update()
{
    //Set healthpack colour to the current level varient
    healthpackAnimator.SetInteger("Level", currentLevel);
}
```

Animation Controller



Healthpack Script: https://pastebin.com/gRKXAG2H

Showcase (4:11): https://youtu.be/lzxFqtqxteU?t=251

Player Controller Script: https://pastebin.com/KqrCVG71

Level Change

A prefab and script was made for the end of the levels when the player has completed the level and needs to progress to the next level.

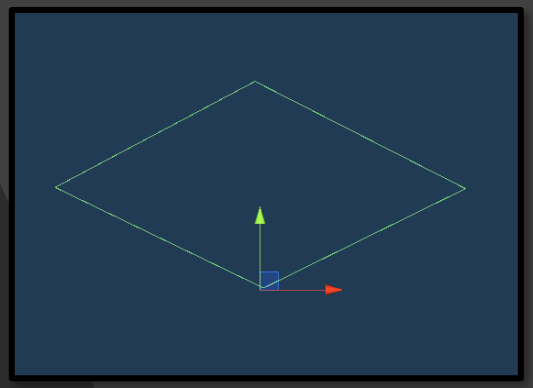
To ensure the player has filled up the colour bar (our level complete indicator) a check is run to see if the colour bar is full, otherwise the player will not be able to progress through.

Currently there is only a debug message that informs the player they need to fill the colour bar, however with further time I would add an in-game notification that tells the player they need to kill more pigments first.

The level that the player moves to is controlled with a public string variable that can be changed per object put in the scene. This means it can be used multiple times in different scenes and link to different scenes as needed.

The YouTube clip shows an orange tile where the collision box is, however this is hidden in-game.

Collision Box



Collision Detection

```
//---------- COLLISION DETECTION ----------
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //If the player collides with the object AND the c
    if (collision.transform.CompareTag("Player"))
    {
        if (player.GetColourFull())
        {
            SceneManager.LoadSceneAsync(newLevel);
        }
        //Otherwise a debug log will be printed
        else
        {
            Debug.Log("Fill the colour bar first!");
        }
    }

}
```

Script use in scene

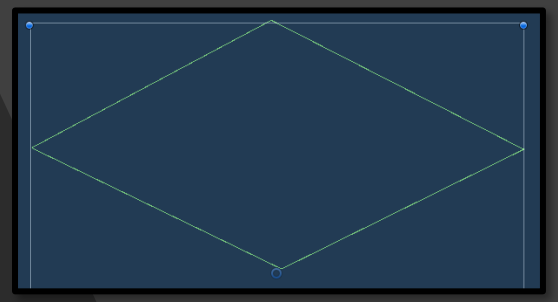
Showcase(4:18): https://youtu.be/lzxFqtqxteU?t=258

Level Change Script: https://pastebin.com/Bw2Hs67C

Room Teleport

As the isometric viewpoint can get confusing at times I decided to split each room up so only one room is visible at a time. They are still in the same scene and needed a way to move between rooms. I created a teleport script similar to the level change script. The script also changes the players room number so different enemies activate and deactivate as required.

Collision Box



Collision Detection

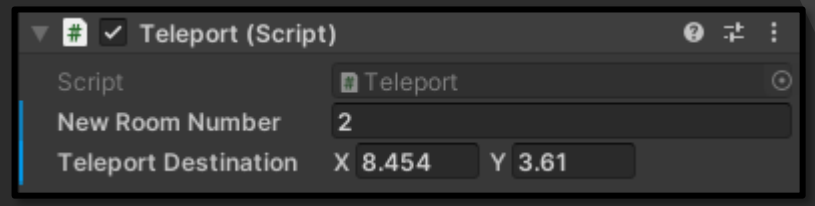
```
//---------- COLLISION DETECTION ----------
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    //If the player collides with the object the players pc
    //The room number and player start position (for checkp
    if (collision.transform.CompareTag("Player"))
    {
        //Update player position
        player.transform.position = (teleportDestination);

        //Update room number
        player.SetRoomNumber(newRoomNumber);

        //Set player checkpoint
        player.SetStartPos(teleportDestination);
    }
}
```

These teleports are placed in door ways and room change points and have the new room number and teleport vector added.

A checkpoint is also updated in the player script to spawn the player back here when they die. This is not used in the final build as the game over screen is now shown.

Looking back I believe I could have simplified this and made it easier to use by calculating the position using a transforms position in the prefab.

Script use in scene



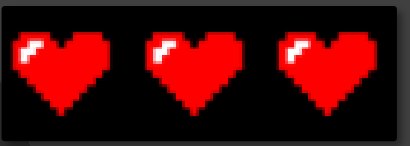
Showcase (4:33): https://youtu.be/lzxFqtqxteU?t=273

Teleport Script: https://pastebin.com/xiPnn4yc

Heart System

Old Healthbar

Old hearts system

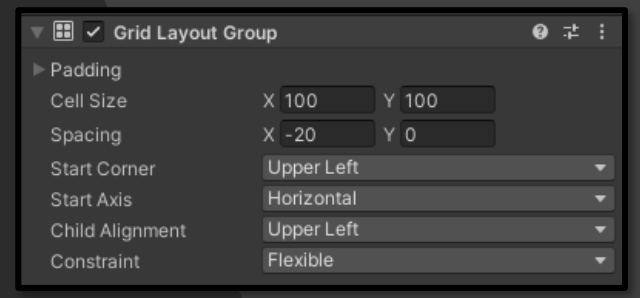


I originally created a basic healthbar that used a slider to update the player health. This was later scrapped for a hearts system that I created with some temporary assets while art was created.

Once artwork was finished for the hearts, I realised I needed to rework the health system up as the artists had created empty, half and full hearts. I needed a system that would show empty heart icons when health had been lost.

I used a YouTube tutorial *(Muddy Wolf Games, 2020)* while creating the reworked system.

Prefab setup

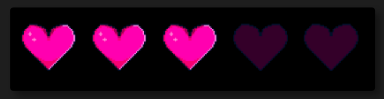


This new system can be updated by calling the function through the player controller script. It can be used to remove or add health as needed.

First the script deletes any existing hearts in the UI. Then it loops through the players max health and creates either empty or full hearts depending on the players current health.

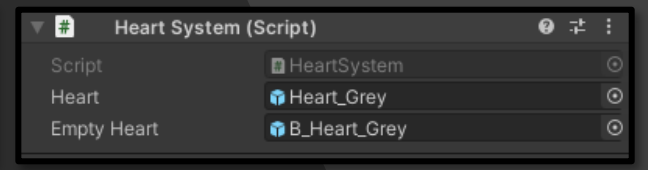
I used a grid layout to create an area where the hearts would be positioned.

Finally as there were different variants for each level I made the heart and empty heart prefabs public so that they could be adjusted with the correct colour for each level.



Destroying old hearts

```
//Loops through each object in the curr
foreach (Transform child in transform)
{
    Destroy(child.gameObject);
}
```



Creating new hearts

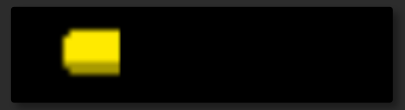
```
//Loops for max health and creates different hearts depending on how much health the player has
for (int i = 0; i < maxHearts; i++)
{
    //Creates as many full hearts as the player has health
    if (i + 1 <= hearts)
    {
        GameObject newHeart = Instantiate(heart, transform.position, Quaternion.identity);
        newHeart.transform.SetParent(transform); ;
        newHeart.transform.localScale = new Vector3(1, 1, 1);
    }
    //Any extra spaces left are filled with empty hearts to show the players max health.
    else
    {
        GameObject newHeart = Instantiate(emptyHeart, transform.position, Quaternion.identity);
        newHeart.transform.SetParent(transform);
        newHeart.transform.localScale = new Vector3(1, 1, 1);
    }
}
```

Showcase (4:45): https://youtu.be/lzxFqtqxteU?t=285

HeartSystem Script: https://pastebin.com/zNGX954Y

Colourbar

Old colourbar



Final colourbar

As a temporary colourbar I created a script that controlled a UI slider on the game hud.

The final colourbar is used to show how much colour the player has collected in the level so far. The player will be unable to progress until they have filled this bar. This mechanic is controlled via the player controller and HUD prefab.

I created two images on the HUD, one for the inside filler and one for the border. These had different animations depending on the current level so I updated these using an integer in the animation controller.

The colour can be updated by calling the AddColour function in the player controller. Due to the angled shape of the colourbar sprites, I made the first and last segments of the colour bar slightly larger using an if statement as they were difficult to see with the standard increase. Once the colour is full the colourFull boolean is set to true which allows the player to progress the level. The colour is updated every time an enemy pigment dies and the DestroyPigment is run in their script.

The colourbar fill amount is updated in another function to keep it up to date. This function also disables the colour bar if the player is in the boss level and enables it if they are in a standard level.

Add Colour method

```
//Adds to the colour amount (can  be called from other scripts)
4 references
public void AddColour(float colour)
{
    //As the colour bar has an angled design, making each increm
    //This if statement corrects this by making the first and la
    if (currentColour < 0.1f)
    {
        currentColour = 0.1f;
    }
    else if (currentColour > 0.9f)
    {
        currentColour = 1f;
    }
    else if (currentColour >= 0.1f && currentColour <= 0.9f)
    {
        currentColour += colour;
    }

    //This checks if the colour has been filled and allows the p
    if (currentColour == 1)
    {
        colourFull = true;
    }
```

Update colourbar

```
//Updates the colour bar UI
1 reference
private void UpdateColourBar()
{
    //Updates the colourbar with the current colou
    colourFill.fillAmount = currentColour;


    //If the current level is the boss level, the
    if (currentLevel == 4)
    {
        colourBorder.enabled = false;
        colourFill.enabled = false;
    }
    //Any other levels the colour bar is visible.
    else
    {
        colourBorder.enabled = true;
        colourFill.enabled = true;
    }
}
```

Update Animations

```
//Updates colour animation variables to ensure the correc
colourBorderAnimator.SetInteger("Level", currentLevel);
colourFillAnimator.SetInteger("Level", currentLevel);
```

Called in enemy

```
public void DestroyPigment()
{
    player.AddColour(colourValue);
    Destroy(gameObject);
}
```

Prefab Setup

Colourbar Script (Unused): https://pastebin.com/YtKX2meW

Player Controller Script: https://pastebin.com/KqrCVG71

Showcase (4:53): https://youtu.be/lzxFqtqxteU?t=293

Boss Healthbar

Prefab Setup

The boss healthbar functions similarly to the colour bar using two images (Inside fill and border).

The boss' health is updated in the take damage script which also triggers the inside fill to update to show the new health.

As with the colour bar the first and last increments were hard to see so I made them slightly larger using an if statement.

If the boss dies or the player is on a different level, the boss healthbar is hidden from view.

Take Damage / Update Healthbar

```
public void TakeDamage(int damage)
{
    //Plays a damage effect
    SFXManager.PlaySound("BossHurt");

    animator.Play("Hurt");
    hurt = true;

    //Lowers the lizard wizards health
    currentHealth -= damage;

    //Updates the healthbar
    UpdateHealth(currentHealth / maxHealth);
}
```

Update Health fill amount

```
public void UpdateHealth(float health)
{
    //When updating the healthbar the first and last incr
    //This if statement calculates when it is the first o
    //Otherwise it will add up as normal.
    if (healthFill < 0.06f)
    {
        healthFill = 0.06f;
    }
    else if (healthFill > 0.94f)
    {
        healthFill = 0.94f;
    }
    else if (healthFill >= 0.06f && healthFill <= 0.94f)
    {
        healthFill = health;
    }
}
```

Toggle if UI is active

```
private void ShowUI()
{
    //If the lizard wizard is destroyed the health bar will
    if (destroyed)
    {
        bossHealthBorder.enabled = false;
        bossHealthFill.enabled = false; ;
    }
    else
    {
        //If the boss level is active, the boss is activated
        if (player.GetCurrentLevel() == 4)
        {
            bossHealthBorder.enabled = true;
            bossHealthFill.enabled = true;
        }
        //Otherwise the boss and healthbar are deactivated.
        else
        {
            isActive = false;
            bossHealthBorder.enabled = false;
            bossHealthFill.enabled = false; ;
        }
    }
}
```

Trigger Take Damage

```
if (collision.transform.CompareTag("Lizard Hitbox"))
{
    //Sound effect plays
    SFXManager.PlaySound("Boss");
    lizardWizard.TakeDamage(damage);

    //Projectile is destroyed
    Destroy(gameObject);
}
```

Update Healthbar

```
private void UpdateHealthBar()
{
    bossHealthFill.fillAmount = healthFill;
}
```

Showcase (3:49): https://youtu.be/lzxFqtqxteU?t=229

Lizard Wizard Script: https://pastebin.com/sKy41jgt

Camera Follow

I created a camera follow script so the camera would follow the player around the level automatically.

Using a blog project from the Unity website *(Hinton-Jones, A. 2019)* I was able to create a script that makes the camera smoothly follow a target object around the scene.

I made this toggleable so that we could either change it per scene or even within script if the camera needed to stop for any reason. This was used for the boss level where I decided the player needed to see the whole level at once instead of following the player.

Follow Function

```
private void Follow()
{
    //Sets the target position with an offset if added in the scene
    Vector3 targetPosition = target.position + offset;

    //Smooths the position so its not as jarring for the player
    Vector3 smoothPosition = Vector3.Lerp(transform.position, targetPosition, smoothAmount * Time.fixedDeltaTime);

    //Sets the camera position to the new smooth position
    transform.position = smoothPosition;
}
```

Toggle Follow

```
//If the follow
if (follow)
{
    Follow();
}
```

Showcase (0:07): https://youtu.be/lzxFqtqxteU?t=7

Camera Follow Script: https://pastebin.com/7VgFhKxv

Hide Tilemaps

A very simple tilemap hiding script was made to stop the collision sprites showing in-game.

This script gets the tilemap renderer and disables it on start-up so when the game launches it no longer renders.

```
public class HideTilemaps : MonoBehaviour
{
    //---------- DECLARE VARIABLES ----------
    private TilemapRenderer tilemapRenderer;
    Unity Message | 0 references
    private void Start()
    {
        //Find tilemap renderer component
        tilemapRenderer = GetComponent<TilemapRenderer>();

        //Disabled the tilemap renderer
        tilemapRenderer.enabled = false;
    }
}
```

Hide Tilemaps Script: https://pastebin.com/btN2gaQq

Illustration Manager

The artists had created various illustrations to use between levels such as introductions and game over screens. I decided to create a script to control these screens moving to the next scene.

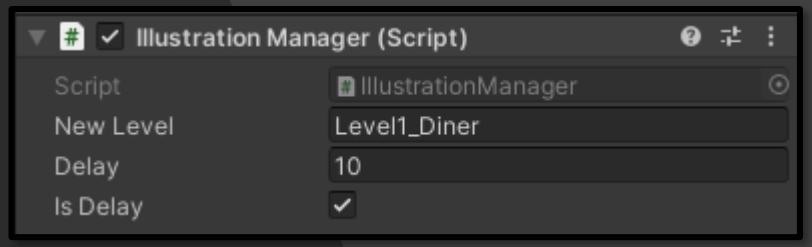
The script is applied to the background and works by either a delay or a button press as decided with public variables. A boolean toggle decides which version the scene will use.

The delay version uses Time.deltaTime to count down a float value and when it reaches 0 will move to the next scene.

The button press will wait for the specified button press before continuing.

I had to load the levels asynchronously due to issues with the enemies moving much faster with the standard load method. This has caused an issue where the same scene cannot be loaded twice that I have been unable to fix so far.

Public Variables / Script Setup



Set up delay

```
private void Start()
{
    //Sets delay
    currentDelay = delay;
}
```

Move to next scene

```
private void Update()
{
    //Counts down the delay
    currentDelay -= Time.deltaTime;

    //Toggle so if the menu is set to use the delay
    if (isDelay)
    {
        if (currentDelay <= 0)
        {
            SceneManager.LoadSceneAsync(newLevel);
        }
    }
    //Otherwise it will wait for the user to press E
    else if (Input.GetKey(KeyCode.Return))
    {
        SceneManager.LoadSceneAsync(newLevel);
    }
}
```

Showcase – Diner Intro (5:10): https://youtu.be/lzxFqtqxteU?t=310

Showcase – Game Over (0:34): https://youtu.be/lzxFqtqxteU?t=34 Illustration Manager Script: https://pastebin.com/tFaWJiKe

Video Manager

We planned to have cutscenes in the game to develop the story outside of the main gameplay. These would be created in video form so I created a script that would move to the next scene once the video has ended.

To do this I took the frame count of the video in the video player and updated the current frame as the video played. Once the current frame matched the frame count the next scene would be loaded.

To get the video to play in the scene I watched a YouTube tutorial *(Vegas, J. 2020)* to create a render texture, raw image canvas and video player game object that played any video I needed implemented.

This combination will allow us to play any number of cutscenes in various positions in the game.

Get video player and frame count

```
Unity Message | 0 references
private void Start()
{
    //Find video player
    videoPlayer = GetComponent<VideoPlayer>();

    //Set the frame count of the video
    //Some videos tested got stuck on the last coup
    frameCount = videoPlayer.frameCount - 2;
}
```

Get current frame and change scene

```
Unity Message | 0 references
private void Update()
{
    //Update which frame is currently playing
    currentFrame = videoPlayer.frame;

    //Once the last frame is played the next scene is loaded.
    if (currentFrame == frameCount)
    {
        SceneManager.LoadScene(nextScene);
    }
}
```

Showcase (5:19): https://youtu.be/lzxFqtqxteU?t=319

Video Manager Script: https://pastebin.com/ZwXeQ36N

Mouse Position (Unused)

As there was a possibility of us creating a ranged attack for the player later down the line I created a script that tested tracking the mouse position relative to the world that could be used to aim the ranged weapon of choice.

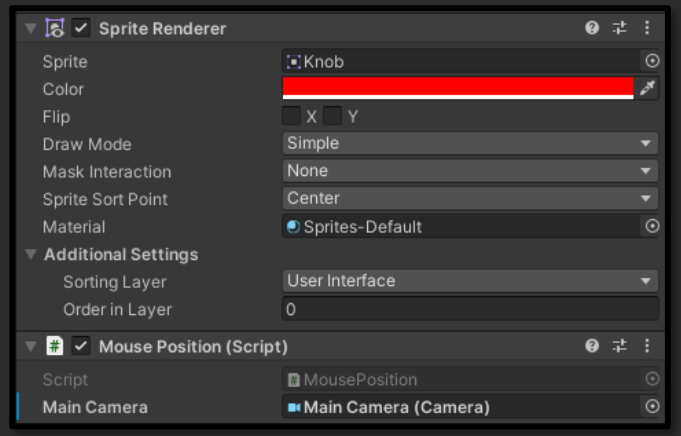
A mouse target prefab was used as a crosshair to visualise the mouse position being updated in real-time.

This did go unused in the final build, however a toned down version was used to update the players attack position relative to the mouse position.

Mouse Position

```
//---------- DECLARE VARIABLES ----------
public Camera mainCamera;
private Vector3 mouseWorldPos;

Unity Message | 0 references
private void Update()
{
    //Creates a vecttor from the mouses input
    mouseWorldPos = mainCamera.ScreenToWorldPoint(Input.mousePosition);
    //Sets the Z position
    mouseWorldPos.z = 0f;

    //Moves the object to the mouses position (Was planned to use as a cr
    transform.position = mouseWorldPos;
}
```

Mouse Target Prefab



Showcase (6:02): https://youtu.be/lzxFqtqxteU?t=362

MousePosition (Unused): https://pastebin.com/GBWi1kXE

Level Design – Boss

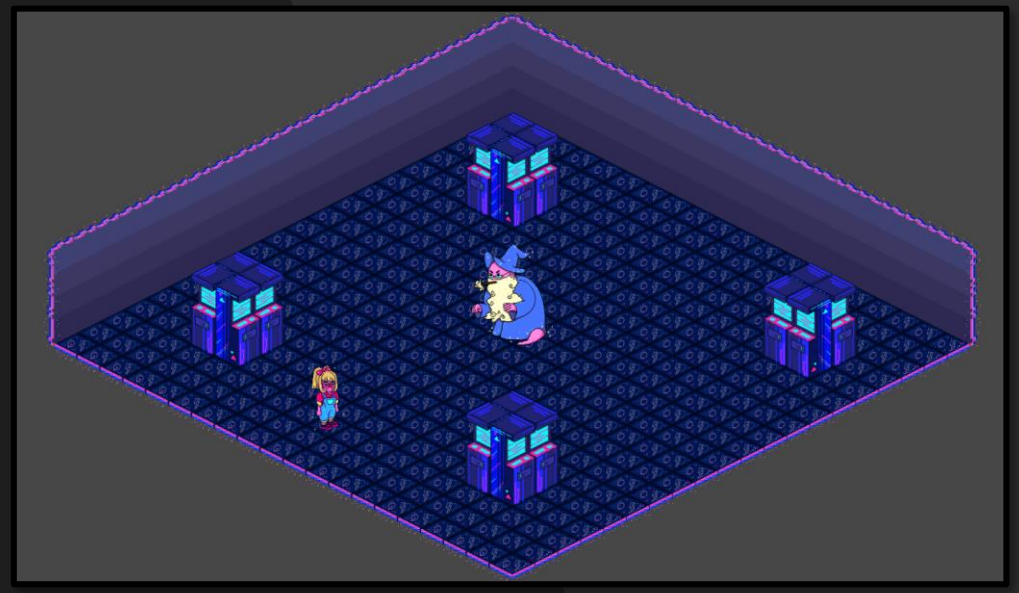
Outside of scripting, I also designed the level for the boss fight. The boss was designed to stand still so I created a square room where the boss would be placed in the centre. This gave the boss adequate space to shoot projectiles, spawn pigments and fire the shockwave attacks.

I wanted to give the player something to hide behind to block the enemy attacks other than hitting them back or dodging with the dash. I originally had arcade machines however this caused collision issues due to the isometric viewpoint with projectiles seemingly going straight through. I changed these to the dance mats and these will destroy any projectiles that collide with them.

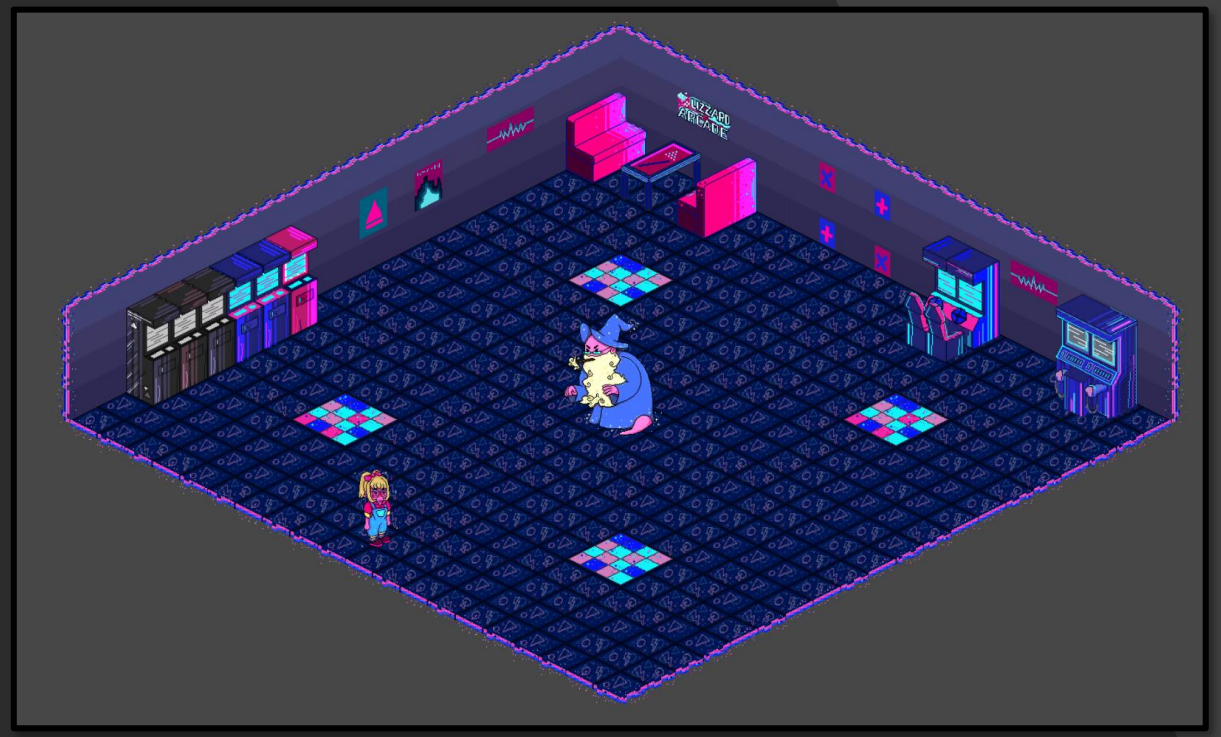
I also added some decoration along the wall using the furniture and wall detail tilemaps created earlier.

I ensured these were not placed too close to the main battle area so the player was unlikely to collide and get stuck on objects.

Initial Design



Final Design



Testing

After completing v1.0 of our game build, we started testing both internally and sharing the game with friends and other students. I created a table to collect some test data and find any bugs we could fix.

This of course highlighted some issue areas as well as some bugs that were affecting gameplay. Due to time constraints we were unable to fix every bug that appeared, however I troubleshooted the ones that I could in the time we had left.

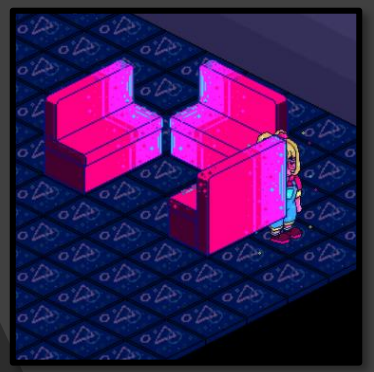
Issue/Bug Description	Location	How Consistent?	Impact on game
Dani can “stand” on the drinks counter	Diner – Room 1	High	Low
Stuck on diner loading screen	Diner – Intro Screen	Low	High
Bottom pigment in diner room 2 can’t hit player as projectiles are blocked by table	Diner – Room 2	High	Medium
Dani can continue to attack after dying, plays attack sounds	Diner, Arcade, Boss	High	Medium
Standard pigments don’t attack continuously if the player stands on top of them	Diner, Arcade, Boss	High	Medium
Dani can stand behind the booth seating	Diner, Arcade, Boss	High	Low
Some exits to each room are difficult to see, especially those in the diner as floor tiles can disappear into the background.	Diner, Arcade	High	Medium
After restarting after dying the player and music plays weird and takes a few seconds to go back to normal.	Diner, Arcade, Boss	Low	Medium
Pigments, healthpacks, colourbar take a bit to update to the correct animation	Diner, Arcade, Boss	High	Low
Was able to leave the diner without completing colour bar	Diner	Low	High
Projectiles don’t disappear when leaving a room	Diner, Arcade	High	Medium
Projectiles go through the arcade machines	Arcade	High	Medium
Player hitbox too large was getting hit by projectiles not touching them	Diner, Arcade, Boss	Medium	Medium
Pigments never spawned in boss stage so game could not progress	Boss	Low	High
Volume slider doesn’t work	Options Menu	High	Medium
During the boss fight if you reflect the projectiles to early they go through the boss without damaging him.	Boss	Low	Low
Some enemies dying increase the colour bar by varying amounts	Diner, Arcade, Boss	Medium	High

Iteration – Bug / Feedback Changes

In the upcoming pages I have discussed these issues and feedback showing any any fixes I made to improve the final build.

Player Sprite Render Order

There is currently an issue where the player clips through furniture objects when they get close to the edges of objects. This is due to the isometric viewpoint which I had issues with solving. I tried a few workarounds but was unsuccessful in fixing the issue. As this is a visual issue only I did not waste any more time on this and moved to more gameplay focused issues.



Stuck on diner loading screen

One user's game was stuck on the diner loading screen where they were unable to move on to the next screen or return to the main menu. I believe this is due to the asynchronous loading method used to ensure the timing of the game works as expected. This does not happen using the standard loading method, however this caused the game speed to be much faster than intended. This bug only occurs when the player tries to play the game after returning to the main menu from the pause menu.

Volume Slider and Fullscreen button

In the options menu there is a volume slider and full screen toggle, these do not currently have any impact on the game itself as they have not been scripted into the game. With further time I would have liked to have implemented these options but decided to focus on the gameplay over these.

Iteration – Bug / Feedback Changes

Dani can still attack after death

After Dani dies and is playing the death animation, she can still attack and damage enemies, even killing them if they get low enough. To fix this I used an if statement that checks if the player is alive before allowing them to attack.

```
//Checks if the player is alive
if(player.GetIsDead() == false)
{
```

Projectiles don't disappear when leaving a room

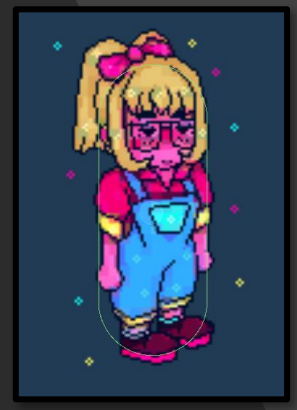
The projectiles created by the ranged enemies could travel through the collisions in the diner and arcade level. This was a small fix that required the collision tilemaps on each to be given the collisions tag that had been missed out. This deleted the projectiles when they collided with the collision tilemap, now working as expected.

Projectiles can move through the arcade machines

This issue is due to the isometric viewpoints effect on collision detection as the collisions are based on the floor tiles not the furniture itself the projectiles can pass through the upper parts of the machines. I have not fixed this in the final build as it would require me to rework the collision system which would take too much time. I would create a separate collision system for movement and projectile detection that would fit any furniture placed in the scene.

Projectiles hitting the player when they don't touch them

This was a minor issue with the players damage hitbox being too large. I changed the collider from a box collider to a capsule collider which better fit the player sprite.



Iteration – Bug / Feedback Changes

Melee pigments don't attack continuously

In the current state, melee pigments wont attack continuously if the player stands on top of them. This means the player can stay on the enemy without taking damage and kill it with ease. As with the other collisions I was unable to rework them fully to fix these sorts of issues however I would create a toggle that controls whether the enemy should attack or not and wait for the player to stop colliding before stopping them from attacking.

Boss Projectiles go through the boss if hit early

Another collision detection issue appears if the player hits projectiles back at the boss early, just after they spawn. This causes some projectiles to not deal damage to the boss and pass straight through. The current collision detection checks when the object enters the collision box which does not update if the projectile never leaves as it does when the player deflects it early. This would be fixed with a collisions overhaul that would not fit in the project timescale.

Pigments never spawned in boss stage

In a rare case, one user's game did not spawn the pigments in the pigment stage so the player was left waiting unable to progress. This seems to be a very rare case as I have not been able to replicate it. The boss stages have been updated for balancing and may have fixed this rare issue.

Player / Music glitching on restart

Another rare issue that affected one users game is the player movement and music glitching after the player had died and restarted the level. As with the pigments not spawning I was unable to replicate this issue and neither was any other users so was unable to diagnose why this happened.

Iteration – Bug / Feedback Changes

Colourbar increasing at random amounts

This issue took a while to fix as there didn't seem to be a pattern to why it was increasing at different amounts. I eventually found a bug with the unity blend trees where it would play multiple animations at once, causing the animation event to trigger multiple times, adding to the colourbar more than once.

I have fixed this by using only one animation for the death of each pigment instead of the blend tree. This does mean they now only face one direction but the colourbar increases as expected. With more time I would like to fix this issue by altering the animation event controls.

Some exits in the diner level are hard to see

Due to the dark background used, some of the diner room exit tiles are very difficult to spot, some users were confused and thought the level was over when there was still another room or two left. I have fixed this by altering the level designs so the entrances and exits are on the white tiles so they contrast to the dark background. These can be seen on upcoming level design pages.

Pigments, healthpacks and colourbar show wrong colour at start

Unfortunately as these are controlled via animations they need to be updated first. This update can be seen in the first couple of frames as the game loads. This doesn't effect gameplay in anyway and is a visual bug only. If the project was continued I could hide this change by implementing a loading screen that waits for the game to load in the background before continuing.

User was able to leave the diner without filling the colourbar.

This bug was a rare occurrence and I was unable to replicate it myself on the new version of the build so could not diagnose why this was happening.

Iteration – Level Design (Diner)

A major feedback area was that the overall game was very difficult in areas while the boss was too easy compared to the pigments. To balance the game better I went around the level improving enemy positions and added more health packs to areas of difficulty. The boss was also balanced to make them much more challenging than before.

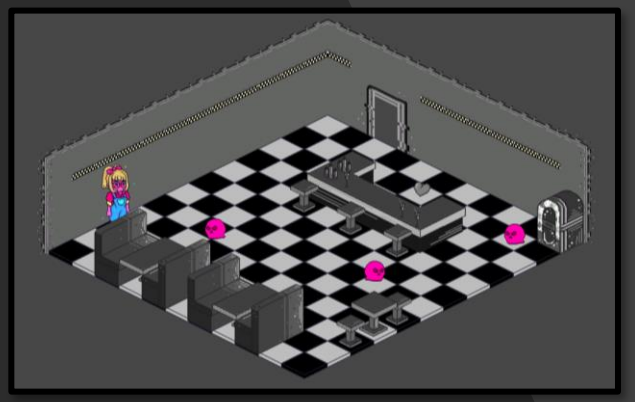
Diner – Room 1

Room 1 of the diner did not require many changes. I repositioned one of the enemies away from the door in case the player returns to the room and didn't kill them before.

I also added some extra collisions to the counter to limit the amount of clipping that could show.

Before (v1.0)

After (v1.1)



Enemies, healthpack and other level specific game objects initially show the wrong colour as they are controlled via scripts

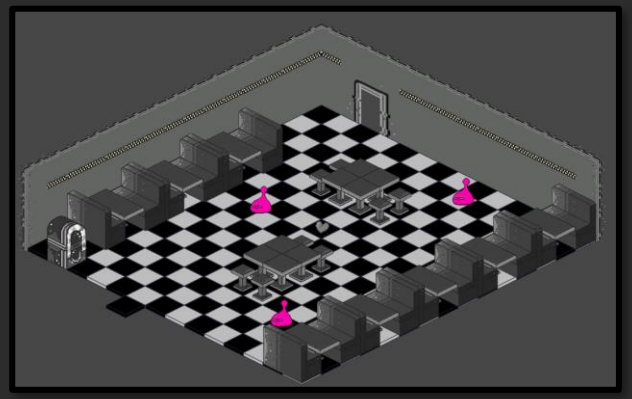
Iteration – Level Design (Diner)

Diner – Room 2

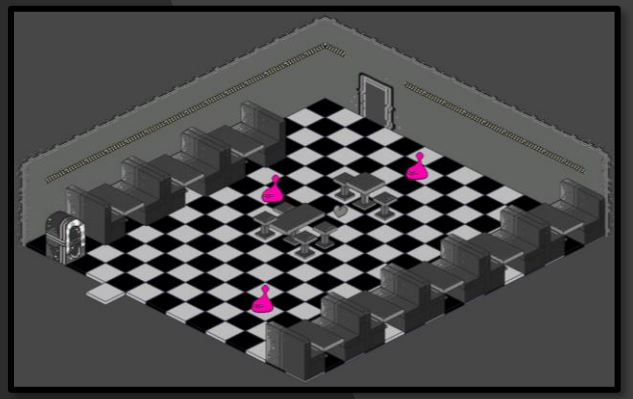
In v1.0 of the build, one of the ranged enemies in this room could not fire projectiles consistently as they kept getting destroyed when they hit the tables above it. I remade the tables slightly smaller and repositioned the enemies so this is less likely to happen.

I also relocated the entrance of the level as the black tile was difficult to see with the black background.

Before (v1.0)



After (v1.1)

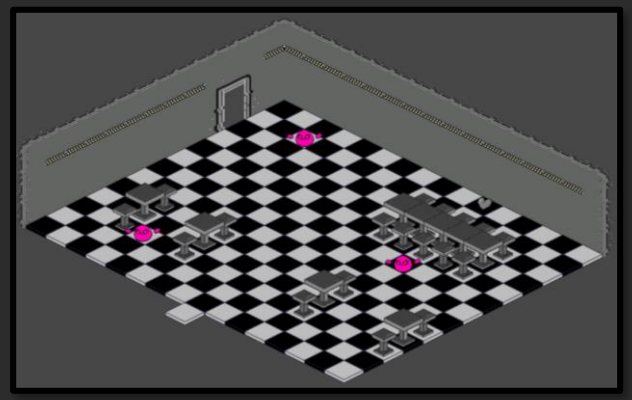


Diner – Room 3

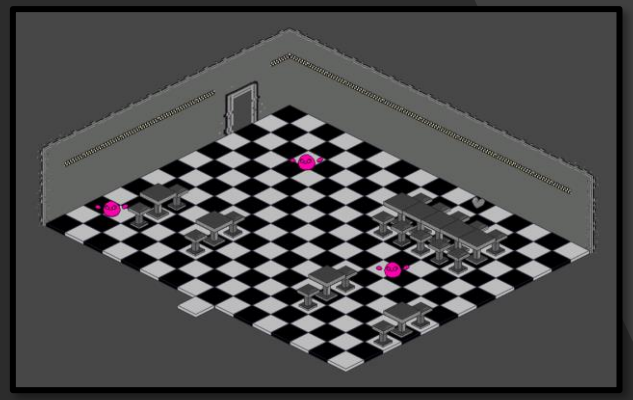
Room 3 introduces the flying enemy types and didn't require many changes.

I repositioned the enemies a bit so they don't attack the player as they move into the room.

Before (v1.0)



After (v1.1)

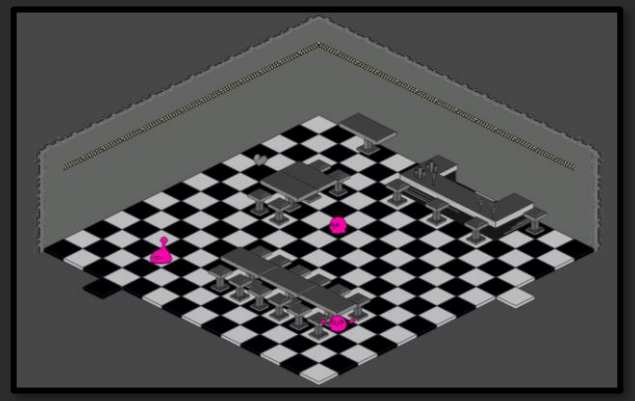


Iteration – Level Design (Diner)

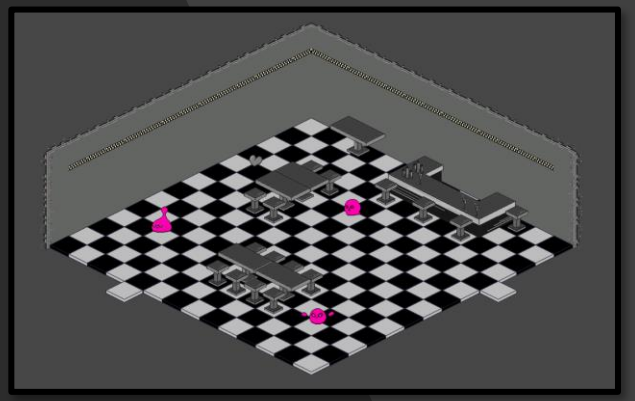
Diner – Room 4

Room 4 felt a bit cluttered so I removed one table set from the level so the player had more space to move around.

Before (v1.0)



After (v1.1)



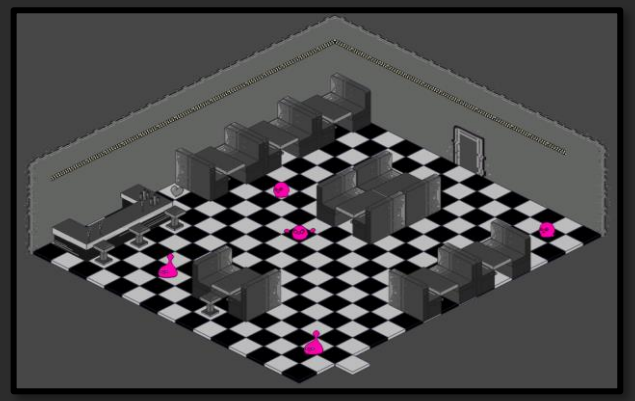
Diner – Room 5

The final room has the most enemies and I felt that there were slightly too many, especially as the player could almost fill the colour bar by then.

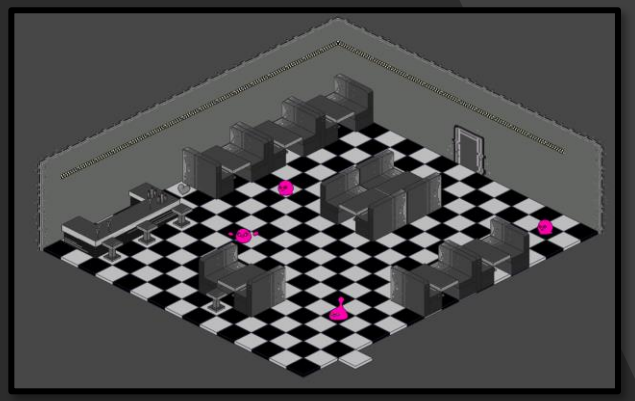
I removed one of the ranged enemies that was rarely fought.

I also repositioned a few enemies so they did not attack the player upon entry to the room and more directly guarded the room exit.

Before (v1.0)



After (v1.1)

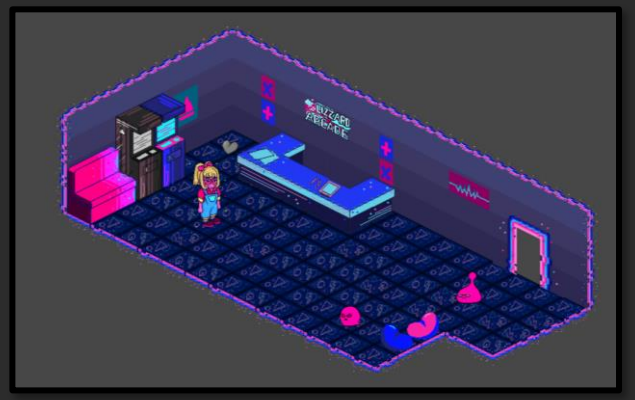


Iteration – Level Design

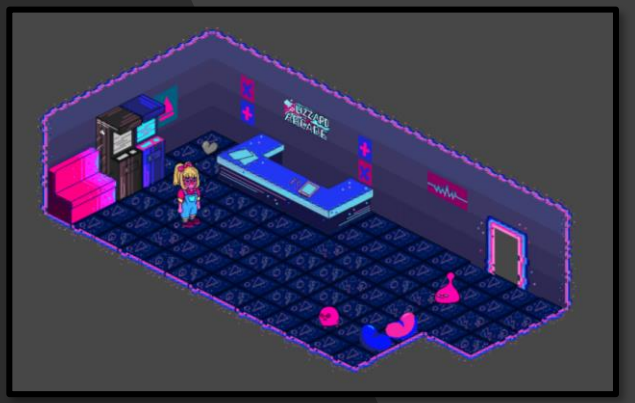
Arcade – Room 1

The first room of the arcade worked pretty well in v1.0 so all I changed was the melee enemy's end position so they don't get quite so close to the players spawn.

Before (v1.0)



After (v1.1)

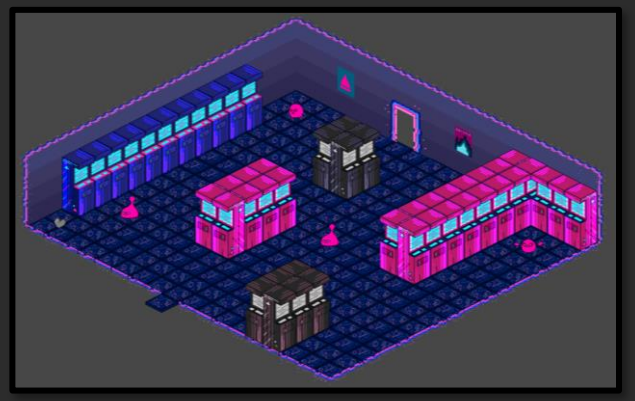


Arcade – Room 2

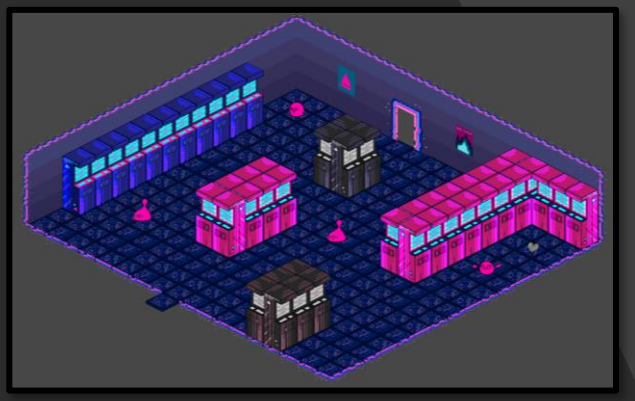
The second room was similar and didn't require many changes.

I moved the healthpack behind the flying enemy in the corner to give the player a challenge before collecting the healthpack.

Before (v1.0)



After (v1.1)



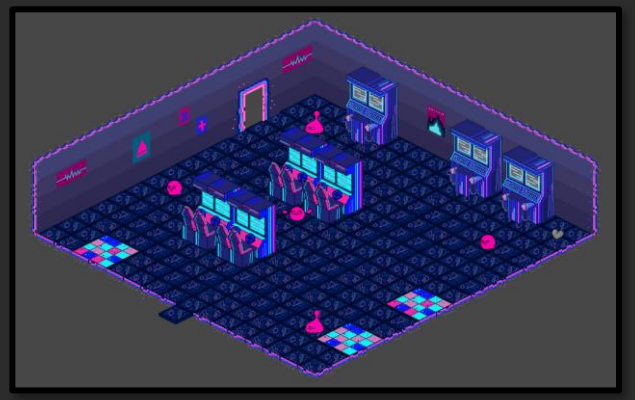
Iteration – Level Design

Arcade – Room 3

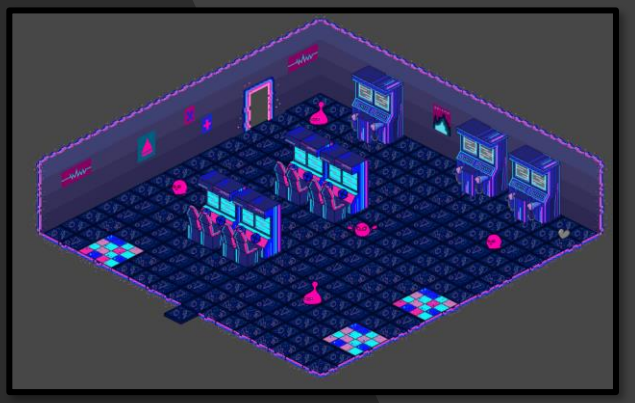
For room 3 I repositioned the enemies away from collisions slightly so they have less issues interacting with them.

I also removed the collisions from the dance mat tiles as users were upset they could not walk on them.

Before (v1.0)



After (v1.1)



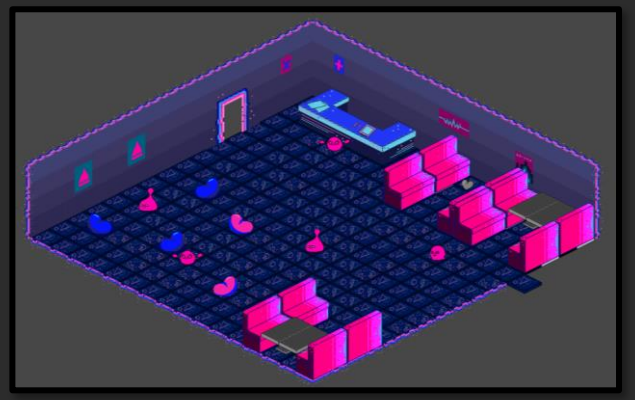
Arcade – Room 4

Room 4 required a visual update by replacing the diner tables with the pool table design for the arcade.

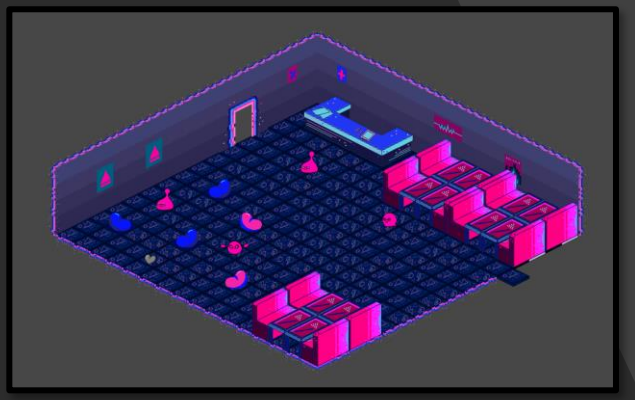
I also moved the enemies away from the starting door to stop the player taking damage on entry.

Finally the healthpack was moved away from the seating to the beanbag area behind some enemies.

Before (v1.0)



After (v1.1)



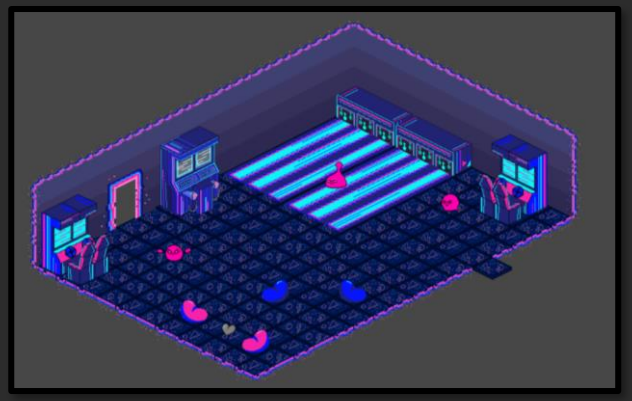
Iteration – Level Design

Arcade – Room 5

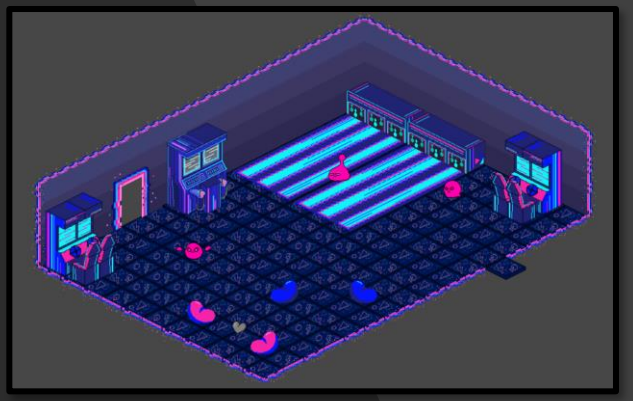
Room 1 of the diner did not require many changes. I repositioned one of the enemies away from the door in case the player returns to the room and didn't kill them before.

I also added some extra collisions to the counter to limit the amount of clipping that could show.

Before (v1.0)



After (v1.1)

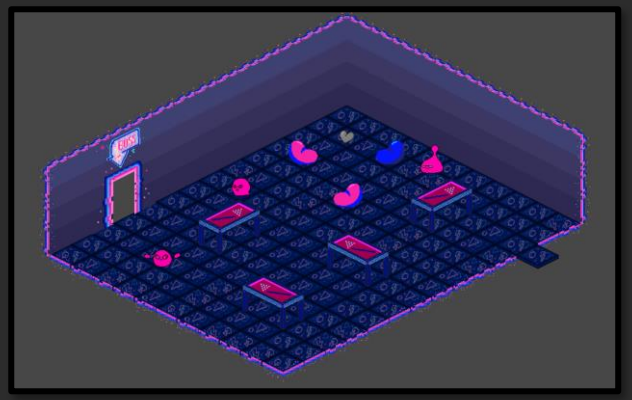


Arcade – Room 6

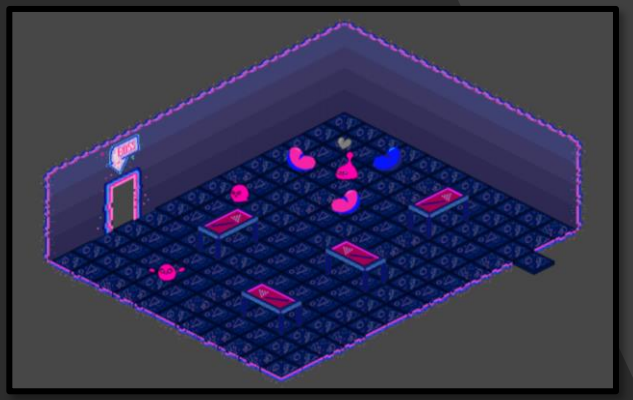
In v1.0 of the build, one of the ranged enemies in this room could not fire projectiles consistently as they kept getting destroyed when they hit the tables above it. I remade the tables slightly smaller and repositioned the enemies so this is less likely to happen.

I also relocated the entrance of the level as the black tile was difficult to see with the black background.

Before (v1.0)



After (v1.1)



Reflection

Having not used Unity for a while it took me a few weeks to get back into the flow. This has resulted in my earlier scripts not being as efficient as some of the later ones. Now that I am more confident with Unity after learning new techniques and coding methods, given the time I would have liked to have gone back to redo some earlier code to improve the functionality.

Learning how to use animation events helped drastically when trying to time certain functions to the animations such as attacking and death animations. These animation events are very useful, and I would have liked to use them more for triggering certain mechanics.

The collision system could do with an overhaul too as with the isometric viewpoint these have been problematic with some collisions not working as expected. Since I have never worked with the isometric viewpoint, I was unsure how to go about creating collisions that fit the viewpoint. If I could redo them I would split collisions up into movement and projectile maps as the projectiles would often appear to go straight through some of the taller furniture objects such as arcade machines as the collision was mapped to the floor tiles.

I would have also liked to do some more level design earlier on as I felt I had little impact on the initial designs. I believe I could have helped more given my knowledge of the enemies and player mechanics that would have made them more efficient and quicker to design.

I also wish I had created more game builds throughout the project as new mechanics were added to allow more of my team members to test the game at various points. The testing we completed at the end of the project was very valuable and brought up issues that I may have been able to fix with more time.

Overall, the project went well both individually and as a group, we were able to deliver a vertical slice of our game concept that had a fairly large scope both on time and to a high standard. There were of course areas that could have been improved to enhance the final build, but we were able to build around our team strengths and adapted when people were struggling with various tasks.

Pastebin Scripts

Here is a complete list of the unity scripts that I worked on. These scripts are fully commented and explain how each script functions in further detail.

- PlayerController: https://pastebin.com/KqrCVG71
- PlayerAttack: https://pastebin.com/8RWjZTjs
- PlayerTakeDamage: https://pastebin.com/MZAJHHLe
- PlayerColour: https://pastebin.com/yxerSG6e
- Greyscale Shader: https://pastebin.com/8uVVAxLa
- SpriteRenderOrder: https://pastebin.com/S6EAPFYu
- EnemyMelee: https://pastebin.com/TsHGr1wd
- EnemyRanged: https://pastebin.com/i79k9MQY
- EnemyRangedProjectile: https://pastebin.com/XCSjGSuL
- EnemyFlying: https://pastebin.com/cgjhHLrw
- LizardWizard: https://pastebin.com/sKy41jgt
- LizardWizardProjectile: https://pastebin.com/PiZg053V
- LizardWizardShockwave: https://pastebin.com/0ifVVYft
- Healthpack: https://pastebin.com/gRKXAG2H
- LevelChange: https://pastebin.com/Bw2Hs67C
- Teleport: https://pastebin.com/xiPnn4yc
- HeartSystem: https://pastebin.com/zNGX954Y
- CameraFollow: https://pastebin.com/7VgFhKxv
- HideTilemaps: https://pastebin.com/btN2gaQq
- VideoManager: https://pastebin.com/ZwXeQ36N
- IllustrationManager: https://pastebin.com/tFaWJiKe
- MousePosition (Unused): https://pastebin.com/GBWi1kXE
- ColourBar (Unused): https://pastebin.com/YtKX2meW

References

- Hinton-Jones, A., 2019. Isometric 2D Environments with Tilemap. [online] Unity Technologies Blog. Available at: <https://blogs.unity3d.com/2019/03/18/isometric-2d-environments-with-tilemap/> [Accessed 28 April 2021].
- samyam, 2020. Making an Isometric Tilemap with Elevations and Colliders in UNITY. [online] Available at: <https://youtu.be/_TY0F7Zm6Lc> [Accessed 28 April 2021].
- Code Monkey, 2020. Character Controller in Unity 2D! (Move, Dodge, Dash). [online] YouTube. Available at: <https://youtu.be/Bf_5qIt9Gr8> [Accessed 28 April 2021].
- Blackthornprod, 2018. HOW TO MAKE 2D MELEE COMBAT. [online] YouTube. Available at: <https://youtu.be/1QfxdUpVh5I> [Accessed 28 April 2021].
- Kee Gamedev, 2020. Unity Simple Sprite Grayscale Tutorial. [online] YouTube. Available at: <https://youtu.be/ktybkQZp2A4> [Accessed 28 April 2021].
- Chris' Tutorials, 2018. Control Sprite Rendering Order (Which 2D Objects Show in Front). [online] YouTube. Available at: <https://www.youtube.com/watch?v=t1UwAGFLmrk> [Accessed 28 April 2021].
- Muddy Wolf Games, 2020. Dynamic Heart System in Unity. [online] YouTube. Available at: <https://youtu.be/ktybkQZp2A4> [Accessed 28 April 2021].
- Vegas, J., 2020. HOW TO EASILY PLAY A VIDEO IN UNITY TUTORIAL. [online] YouTube. Available at: <https://youtu.be/p7iXEZGx2Mc> [Accessed 28 May 2021].