Cookie Crook

Emily Pickhardt, Katie Valentine, Nick Wood, & Aly Shaw

Our Pitch

The dinosaurs never left!

In the next solar system over, dinosaurs evolved and govern their own planet, although they lack one thing.... Cookies.

Play as an extraterrestrial t-rex sent by your civilization to collect Earth’s most delicious treats to return to your planet with.

Player Profiles

Middle age population. Primarily women, but also men

~Retro nostalgic feel
~Quick and easy to play between a busy schedule

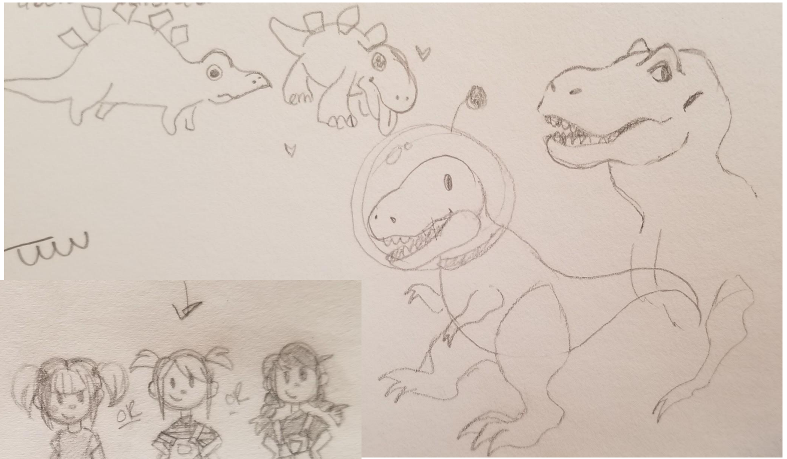
Children ages 6-12

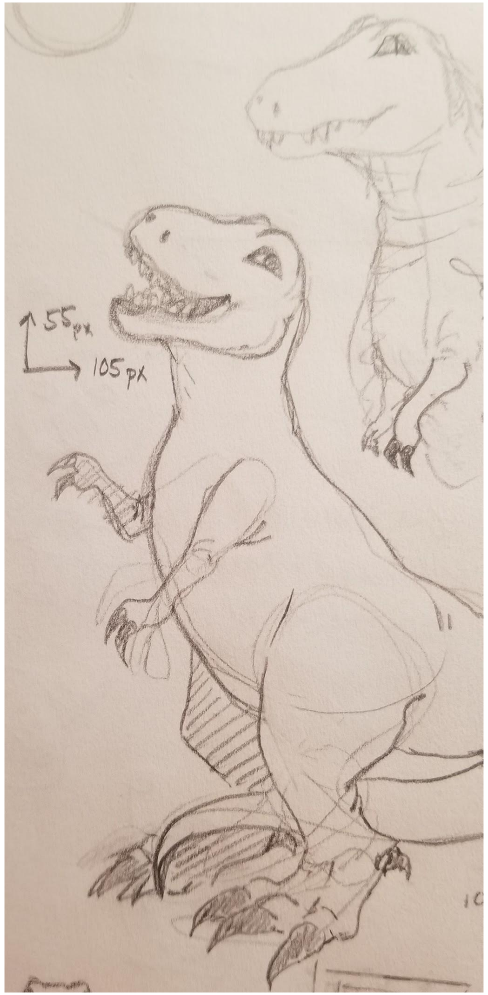
~Very easy to learn

~Great replayability - Won't get bored easily

Can reach many more audiences due to its replayability and simple but fun gameplay

Concept Art





Outline of Experience

- ➔ **Concept and Scope**

  We layed out all the specifics of the game along with how much we're doing before starting on any creation. This included style, rules, scoring, level count, and what milestones we needed to hit at what times.

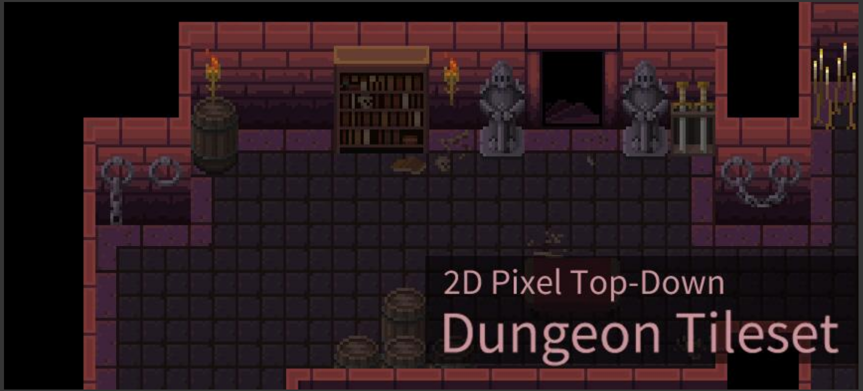
- ➔ **Creation**

  We worked cohesively to ensure that all sprites would work and look correct together. Simultaneously, all programming was being completed with stand in sprites.

- ➔ **Finalization**

  Worked out any kinks and put all assets in place.

Inspiration

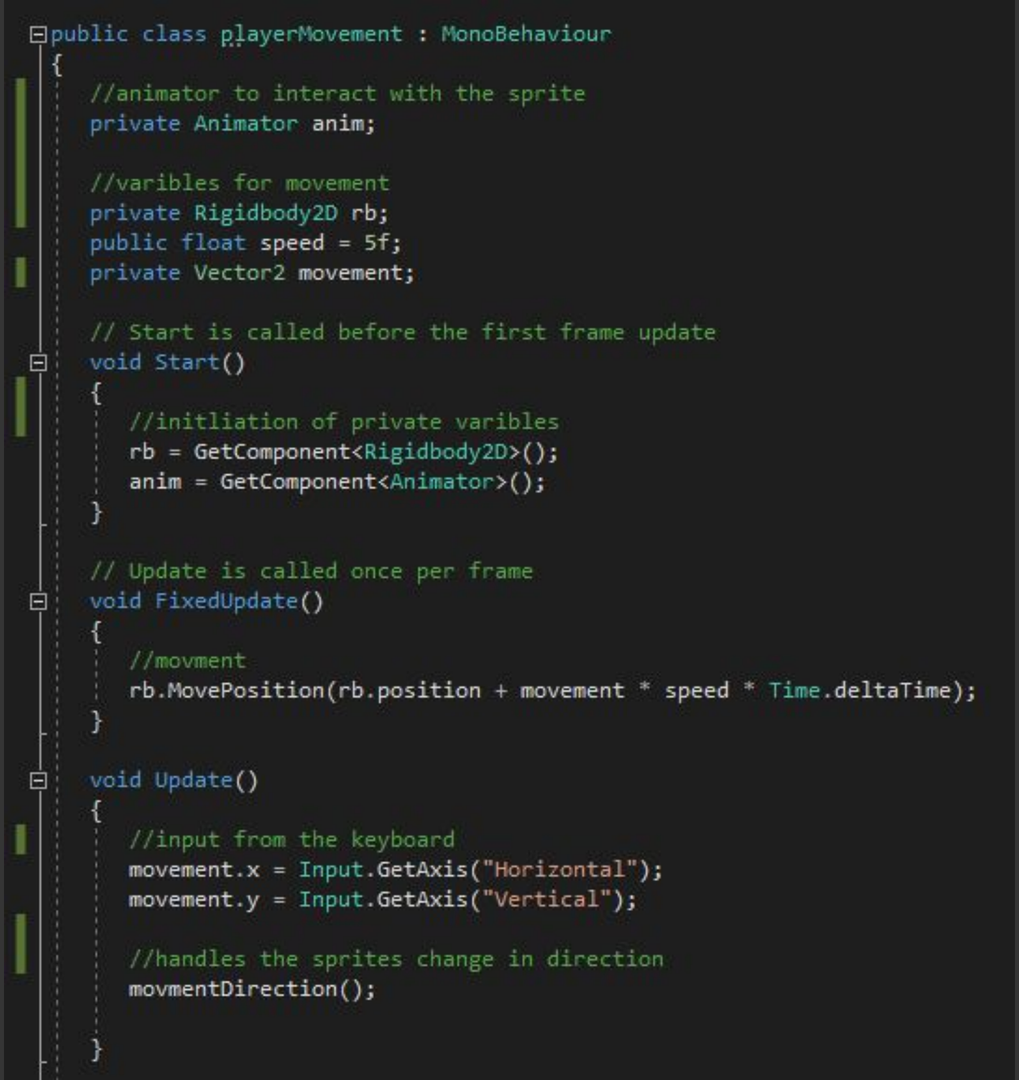


Game Art/Sprite Sheets/UI

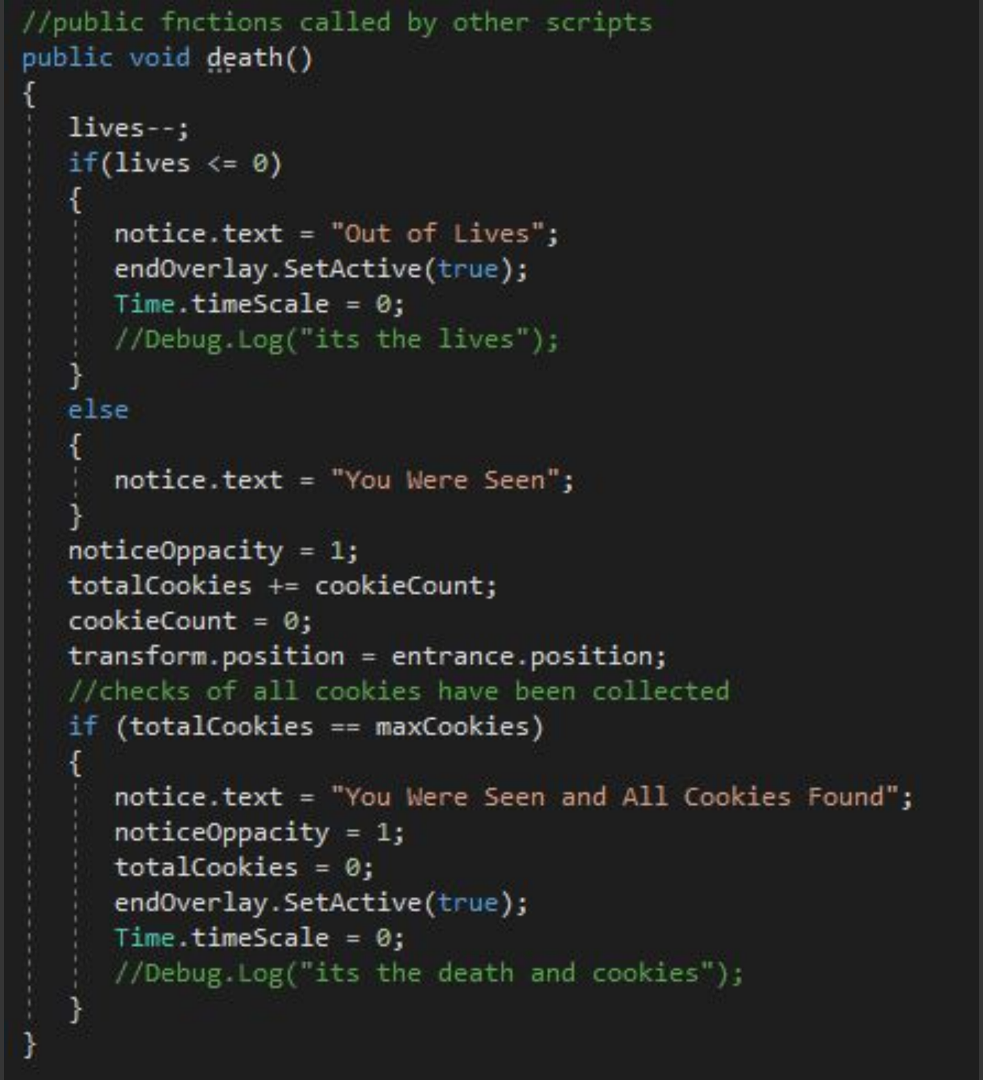
Key Mechanics

- ➔ **Points**
  Players get points for collecting and hoarding cookies, as well as a time bonus.
- ➔ **Support Local Business**
  Play for points and get rewarded with coupons for T-Rex Cookie Kitchen!
- ➔ **Navigation**
  Find your way around the houses to discover more cookies.
- ➔ **Enemies**
  Avoid pathfinding enemies to be able to get more cookies and more points!

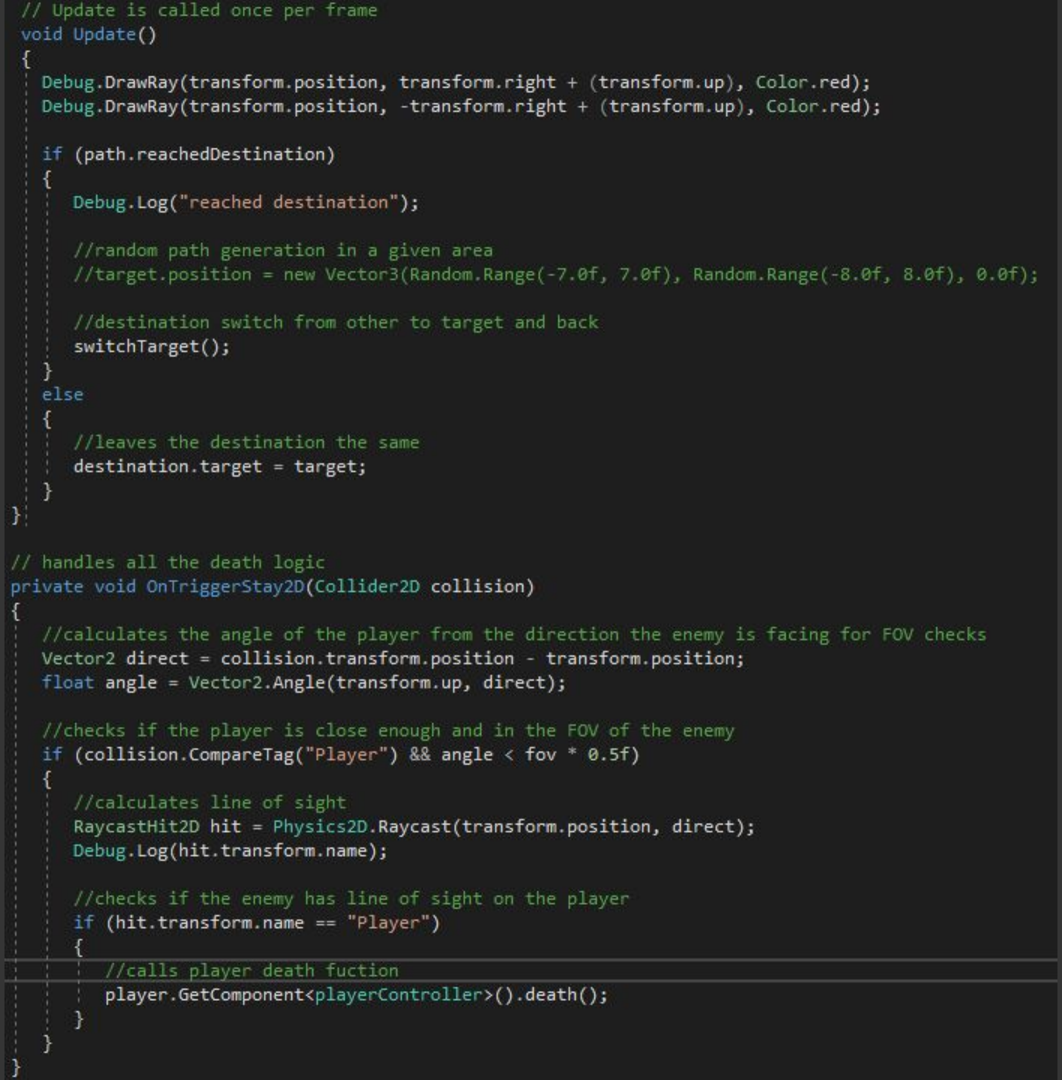
Code



```
public class playerMovement : MonoBehaviour
{
    //animator to interact with the sprite
    private Animator anim;

    //varibles for movement
    private Rigidbody2D rb;
    public float speed = 5f;
    private Vector2 movement;

    // Start is called before the first frame update
    void Start()
    {
        //initliation of private varibles
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
    }

    // Update is called once per frame
    void FixedUpdate()
    {
        //movment
        rb.MovePosition(rb.position + movement * speed * Time.deltaTime);
    }

    void Update()
    {
        //input from the keyboard
        movement.x = Input.GetAxis("Horizontal");
        movement.y = Input.GetAxis("Vertical");

        //handles the sprites change in direction
        movmentDirection();

    }
```



```
//public fnctions called by other scripts
public void death()
{
    lives--;
    if(lives <= 0)
    {
        notice.text = "Out of Lives";
        endOverlay.SetActive(true);
        Time.timeScale = 0;
        //Debug.Log("its the lives");
    }
    else
    {
        notice.text = "You Were Seen";
    }
    noticeOppacity = 1;
    totalCookies += cookieCount;
    cookieCount = 0;
    transform.position = entrance.position;
    //checks of all cookies have been collected
    if (totalCookies == maxCookies)
    {
        notice.text = "You Were Seen and All Cookies Found";
        noticeOppacity = 1;
        totalCookies = 0;
        endOverlay.SetActive(true);
        Time.timeScale = 0;
        //Debug.Log("its the death and cookies");
    }
}
```



```
// Update is called once per frame
void Update()
{
    Debug.DrawRay(transform.position, transform.right + (transform.up), Color.red);
    Debug.DrawRay(transform.position, -transform.right + (transform.up), Color.red);

    if (path.reachedDestination)
    {
        Debug.Log("reached destination");

        //random path generation in a given area
        //target.position = new Vector3(Random.Range(-7.0f, 7.0f), Random.Range(-8.0f, 8.0f), 0.0f);

        //destination switch from other to target and back
        switchTarget();
    }
    else
    {
        //leaves the destination the same
        destination.target = target;
    }
}

// handles all the death logic
private void OnTriggerStay2D(Collider2D collision)
{
    //calculates the angle of the player from the direction the enemy is facing for FOV checks
    Vector2 direct = collision.transform.position - transform.position;
    float angle = Vector2.Angle(transform.up, direct);

    //checks if the player is close enough and in the FOV of the enemy
    if (collision.CompareTag("Player") && angle < fov * 0.5f)
    {
        //calculates line of sight
        RaycastHit2D hit = Physics2D.Raycast(transform.position, direct);
        Debug.Log(hit.transform.name);

        //checks if the enemy has line of sight on the player
        if (hit.transform.name == "Player")
        {
            //calls player death fuction
            player.GetComponent<playerController>().death();
        }
    }
}
```