COLLEGE OF ENGINEERING | **Electrical Engineering and Computer Science**

AR PAC-MAN

Augmented Reality Pac-Man Game developed for iOS/Android using Unity

TIMOTHY BUTLER / NICK THOMAS

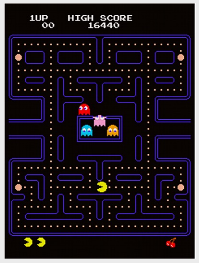
DEVELOPMENT

- Unity software was used for the development of the game, which features AR Foundation, a cross-platform framework that allows you to build augmented reality experiences once, then build for either Android or iOS devices.
- Unity utilizes C# scripting for assistance with game logic and manipulating game objects. This was used to control the flow of the game, the logic of the ghosts, the movement of Pac-Man, etc., and tied into Unity via the development interface.
- Blender was used to create the models for the game as well as the animation of Pac-Man's mouth while moving.
- Royalty free sound effects for the original game were used, and can be found HERE https://drive.google.com/drive/folders/1RdJQ3S0F-iW3wkb2yQbfxGlVvTMS6Gik

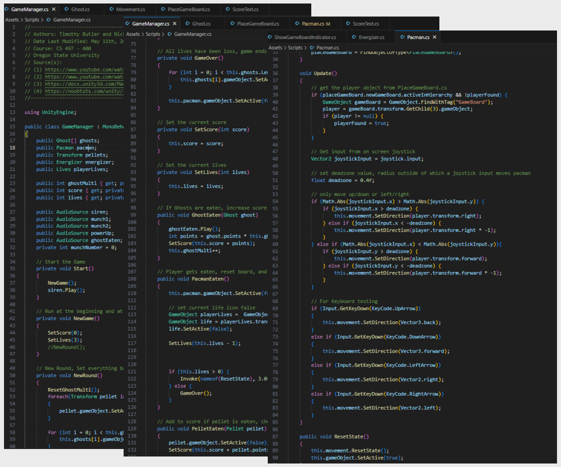
LEARNING EXPERIENCE

- The team gained a tremendous amount of Unity and C# experience, having never used either of them previously. The learning curve of Unity is quite steep, nonetheless we managed to make a lot of progress towards our project plan original goals.
- The team also gained familiarity working on software involving augmented reality, manipulating 2D/3D game objects in 3D world space, and the various positional and rotational transformations that need to be programmed into objects' movement to make the game feel like the original.
- Additionally, we gained more expertise in other development software/tools (Blender, Play Store), and project management tools (GitHub, Trello).

Oregon State University



Original Pac-Man game.



GameManager.cs × | Ghost.cs | Movement.cs | PlaceGameBoard.cs | ScoreText.cs

Assets > Scripts > GameManager.cs

```
//-----------------------------------
// Authors: Timothy Butler and Nicl
// Date Last Modified: May 11th, 2(
// Course: CS 497 - 400
// Oregon State University
// Source(s):
// (1) https://www.youtube.com/wat
// (2) https://www.youtube.com/wat
// (3) https://docs.unity3d.com/Mai
// (4) https://noobtuts.com/unity/
//-----------------------------------

using UnityEngine;

public class GameManager : MonoBehi
{
    public Ghost[] ghosts;
    public Pacman pacman;
    public Transform pellets;
    public Energizer energizer;
    public Lives playerLives;

    public int ghostMulti { get; pr
    public int score { get; private
    public int lives { get; private

    public AudioSource siren;
    public AudioSource munch1;
    public AudioSource munch2;
    public AudioSource powerUp;
    public AudioSource ghostEaten;
    private int munchNumber = 0;

    // Start the Game
    private void Start()
    {
        NewGame();
        siren.Play();
    }

    // Run at the beginning and at
    private void NewGame()
    {
        SetScore(0);
        SetLives(3);
        //NewRound();
    }

    // New Round, Set everything ba
    private void NewRound()
    {
        ResetGhostMulti();
        foreach(Transform pellet i
        {
            pellet.gameObject.SetA
        }

        for (int i = 0; i < this.gl
            this.ghosts[i].gameObj
        }
```

GameManager.cs × | Ghost.cs | PlaceGameBoard.cs | Pacman.cs M | ScoreText.cs

Assets > Scripts > GameManager.cs

```
    // All lives have been loss, game ends
    private void GameOver()
    {
        for (int i = 0; i < this.ghosts.Lei
            this.ghosts[i].gameObject.SetA
        }

        this.pacman.gameObject.SetActive(f
    }

    // Set the current score
    private void SetScore(int score)
    {
        this.score = score;
    }

    // Set the current lives
    private void SetLives(int lives)
    {
        this.lives = lives;
    }

    // If Ghosts are eaten, increase score
    public void GhostEaten(Ghost ghost)
    {
        ghostEaten.Play();
        int points = ghost.points * this.gl
        SetScore(this.score + points);
        this.ghostMulti++;
    }

    // Player gets eaten, reset board, and
    public void PacmanEaten()
    {
        this.pacman.gameObject.SetActive(f

        // set current life icon false
        GameObject playerLives =  GameObje
        GameObject life = playerLives.tran
        life.SetActive(false);

        SetLives(this.lives - 1);



        if (this.lives > 0) {
            Invoke(nameof(ResetState), 3.0
        } else {
            GameOver();
        }
    }

    // Add to score if pellet is eaten, ch
    public void PelletEaten(Pellet pellet)
    {
        pellet.gameObject.SetActive(false)
        SetScore(this.score + pellet.point
```

ShowGameBoardIndicator.cs | Energizer.cs | Pacman.cs ×

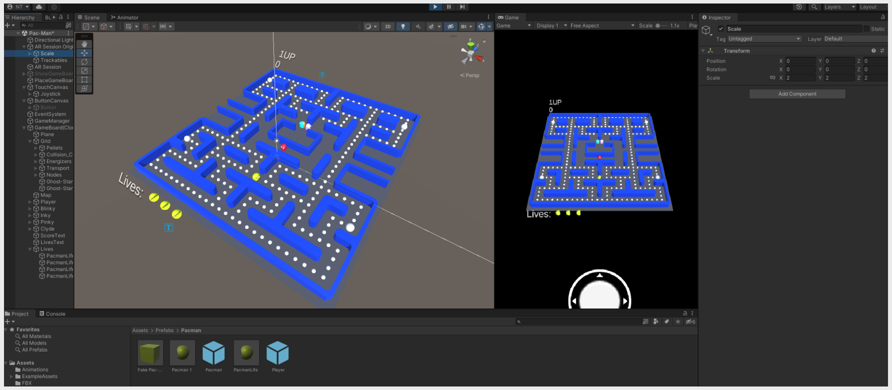
Assets > Scripts > Pacman.cs

```
        placeGameBoard = FindObjectOfType<PlaceGameBoard>();
    }

    void Update()
    {
        // get the player object from PlaceGameBoard.cs
        if (placeGameBoard.newGameBoard.activeInHierarchy && !playerFound) {
            GameObject gameBoard = GameObject.FindWithTag("GameBoard");
            player = gameBoard.transform.GetChild(3).gameObject;
            if (player != null) {
                playerFound = true;
            }
        }

        // Get input from on screen joystick
        Vector2 joystickInput = joystick.input;

        // set deadzone value, radius outside of which a joystick input moves pacman
        float deadzone = 0.4f;

        // only move up/down or left/right
        if (Math.Abs(joystickInput.x) > Math.Abs(joystickInput.y)) {
            if (joystickInput.x > deadzone) {
                this.movement.SetDirection(player.transform.right);
            } else if (joystickInput.x < -deadzone) {
                this.movement.SetDirection(player.transform.right * -1);
            }
        } else if (Math.Abs(joystickInput.x) < Math.Abs(joystickInput.y)){
            if (joystickInput.y > deadzone) {
                this.movement.SetDirection(player.transform.forward);
            } else if (joystickInput.y < -deadzone) {
                this.movement.SetDirection(player.transform.forward * -1);
            }
        }

        // For Keyboard testing
        if (Input.GetKeyDown(KeyCode.UpArrow))
        {
            this.movement.SetDirection(Vector3.back);
        }
        else if (Input.GetKeyDown(KeyCode.DownArrow))
        {
            this.movement.SetDirection(Vector3.forward);
        }
        else if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            this.movement.SetDirection(Vector2.right);
        }
        else if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            this.movement.SetDirection(Vector2.left);
        }
    }

    public void ResetState()
    {
        this.movement.ResetState();
        this.gameObject.SetActive(true);
```

C# Scripting to manipulate game objects and control game logic

FEATURES

- Pac-man Game in the style of the original arcade classic, played through augmented reality using the Unity game engine and AR Foundation, allowing development for both iOS and Android.
- Unity couples with iOS/Android built-in AR tools, and they work together to detect and track position of planes and objects between Unity space and Real-World space.
- Playable demo with functioning ghosts, pellets, power pellets, lives, score, teleport, sound effects etc.



Unity Development Environment, Complete with Game Object Hierarchy, Scene Viewer, Game Simulator, Object Inspector, and File Structure

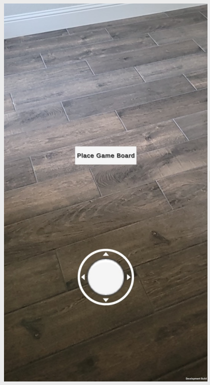
DESCRIPTION

A Japanese company, Namco, created a maze type game in 1980 that soon most people recognized in the list of the greatest games of all time, Pac-Man.

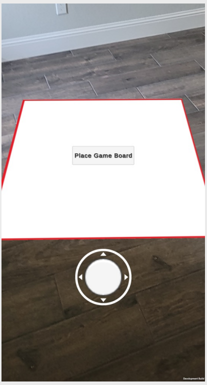
In Pac-Man, you play as a yellow circle shaped character inside the walls of a maze. There are many white dots laid across the maze that the player must pick up to advance to the next level. The player must also avoid getting attacked by four ghosts. If attacked by any of the ghosts, the player will lose a life and the level will restart without replacing the already picked up dots. If the player loses all their lives, then the game is over.

The augmented reality version of this game is much like the classic. In this version, the Unity Engine powers the graphics and game logic, while tapping into the resources inherent to most iOS/Android phones for the AR tracking and object positioning.

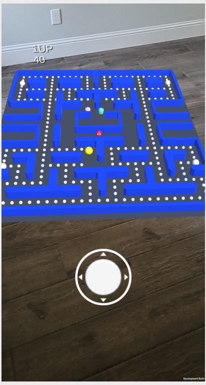
This revitalized classic showcases the enhanced user-experience possible via modern phones and software.



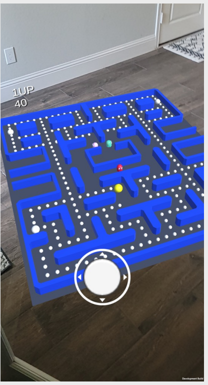
Step 1: Detect Plane as Play Surface



Step 2: Preview footprint of Game Board



Step 3: Press 'Place Game Board' and use Joystick to control Pac-Man.



Step 4: View and Play the game from anywhere while your phone keeps track of the all the 3d Objects!