Writeup Notes

# Grombcross

## Overview

A picross puzzle game made by myself and several Society of Play members themed around Society of Play original characters and inside jokes.

I organized a team of 8 to develop a picross puzzle game using WPF, C#, .NET 6.0, MVVM design principles. I worked on this game after work hours during my internship at the time, using the same tech stack and design patterns as my internship project.

## Puzzle Generation System

1. Each puzzle is generated from just a pair of bitmap images: one black+white (generation image) and one full-color (completed image).
2. The puzzle generation script parses the filename of these bitmap images for information such as puzzle name and puzzle id, and then iterates through the pixels of the generation image to fill a 2D boolean array (representing the puzzle’s solution state).
3. When a puzzle is selected by the user to play, this 2D boolean array is sent as an argument to the game view to generate a game board with the relevant line hints.

An aspect of the puzzle generation system I’m very proud of is allowing the developer or user to easily add, change, or remove puzzles without touching the codebase. The puzzles are generated at runtime using the bmp images in the puzzle data folders (which are intentionally exposed to the user in the distributed build folder).

## Save Data as JSON

The save data format is a dictionary, which is saved into as a json file. The key is the puzzle name (string) and value is the puzzle’s completion state (boolean).

I determined that a dictionary was the perfect data structure for saving game progress because it accommodates for adding, removing, or changing puzzles (either by the developer or by the user via the exposed puzzle data folders) without any issues.

* When loading, all the puzzles in the json file but not in the current game version will be ignored.
* When saving, only the puzzles in the current game version are included in the json file. So, previously-saved puzzles no longer in the game are wiped from the save data.

## UML Diagram

Created a UML to plan out game systems. This helps a lot when setting the project up! Removes a lot of guess work that would otherwise come with programming a project from scratch and keeps the scripts focused.

## Auto-Resizing Board

Game board auto-sizes to fit the window.

When the game view is loaded, the game board’s scale is calculated and set based on window height and width. While in-game, the board scale is recalculated and the board is re-rendered each time the window is resized via a function that listens for the window resized event.

# Project 9th Monday

## Overview

An audio-only horror narrative game following a group of kids on Halloween night looking to raid a haunted house for jewels to fund a video game console purchase.

Organized and led a team of 8 for a 10-day game jam. The team composed of programmers, voice actors, a ui designer, a writer, and a sound engineer. Game ranked highly in game jam.

## Determining Which Speech-to-Text API to Use

Dictation Recognizer, Keyword Recognizer, and machine learning.

## Organizing a Big, Multi-Talented Team

Created a gantt chart and an asset list to communicate assets required and timeline.

# Pizza Delivery Bagel

## Overview

## Simultaneous Android and PC Development (Support different controllers and different playstyles)

Pizza Delivery Bagel supports multiple different control schemes: keyboard, touch, gamepad, and arcade cabinet (for the Dallas Society of Play arcade cabinet version!). When starting a game, the control scheme is set based on the detected platform.

If playing on a PC, the keyboard and gamepad control schemes can be switched between mid-game.

Depending on the control scheme, some control adjustments are made to improve game feel.

* Example: Wall grinding is made easier when playing with a joystick. This is because the wall-grinding speed boost is most effective when moving into a wall at a 45 degree angle, which is easiest to do on keyboard directional arrows.
* Example: When playing on a PC with a gamepad or an arcade cabinet, car acceleration and deacceleration are mapped to separate buttons rather than to the up and down directions. I felt this made the gamepad controls feel significantly more natural.
* Example: When playing on mobile, the car moves in the direction of the virtual joystick, rather than using tank controls. I found that the tank controls felt too awkward and unforgiving on a touchscreen.

## Object Pooling

Utilized object pooling to minimize garbage collection and improve performance. Nothing is destroyed or creating while playing the game. Instead, all objects are created at the start of the game and disabled, enabled, and re-positioned over time as needed!

## Development Tools

I created some custom inspectors with debugging shortcuts.

In-game terminal for debug commands. I used a 3rd party terminal because it provided all the utility I needed and saved time.

## Presentation Skills

Presented progress updates regularly during weekly discord game showcase events. Presented game at two local game conventions for playtesting and feedback.

# The Eaterer

## Overview

A first-person horror game about avoiding unhealthy lifestyle choices.

## Reading Items from Excel Spreadsheet

Items that represent either healthy or unhealthy lifestyle choices. The information for these items was within an Excel spreadsheet with two columns: one for item name and a boolean for whether it was a “good” item or not. From a C# script I read this Excel spreadsheet and used it to generate the items throughout the map. 3D models were stored in an array attached to that script, and using the indexes of the item row to determine which model to spawn the items with.