PARKING TOWERS

Prototype T.D.D.

Overview

Parking Towers is a 2D platformer in which the player, a green cube, must reach a green orb at the end of the level. The player must navigate through a semi-open space riddled with obstacles. They have access to wall jumps and can gain midair jumps by finding powerups throughout the level. Spikes and cars can kill the player, returning them to the last checkpoint reached.

The game is composed of a single level and several menus. Most of the code deals with player movement, player interactions with other objects, and hazard behaviour. A Menu Manager handles the overall game flow.

Requirements

Environment

- A 2D level made of two types of rectangular blocks. It contains spikes, enemy spawners, upgrades, checkpoints, and the victory orb.
- Regular blocks will contain Surface and Wall colliders, which interact with the player differently.
- "PlayerBlocker" blocks will be impassable to the player, but traversable for enemies. They will be a different colour.

Player & Camera Control

- The player will move left and right with the A and D keys. They can jump with the W or Space keys.
- The player can collect jump powerups to unlock additional midair jumps. The player's jumps are reset when they touch the ground.
- While touching a wall, the player can perform a wall jump which will launch them away from the wall and will not consume any midair jumps.
- When the player touches a hazard, they die and have the option to return to the last checkpoint they touched.
- The camera will follow the player with a small buffer; that is, the player will be able to move a short distance in any direction before the camera follows.
- When the player respawns, the camera will snap to their location and attach to the new player object.

Hazards

- In *Parking Towers*, there will be two types of hazards: cars and spikes. Both types of hazards kill the player on contact.
- Cars are moving hazards that inherit behaviour from the "Character" script. This behaviour is shared with the player. It includes certain collision interactions, the ability to die, horizontal movement, and other physics-related behaviour.
- Cars are periodically spawned by car spawners when the player is within a preset range, described by a Physics2D.OverlapBox(). When cars are spawned, their direction of movement will be set to either left or right based on the player's position.
- Cars accelerate along the horizontal axis until they have reached their maximum speed.
- A car can be destroyed in one of two ways: a) upon collision with a wall or hazard, or b) when any object collides with their windshield. The player can destroy a car by landing on its windshield.
- Spikes are hazards attached to surfaces. They are typically stationary, with one exception: a spike in a ceiling may be set to drop when the player passes under it. These spikes will stop falling when they reach a preset target, which is intended to be embedded in another surface.
- DOTween will be used to manage falling spikes.

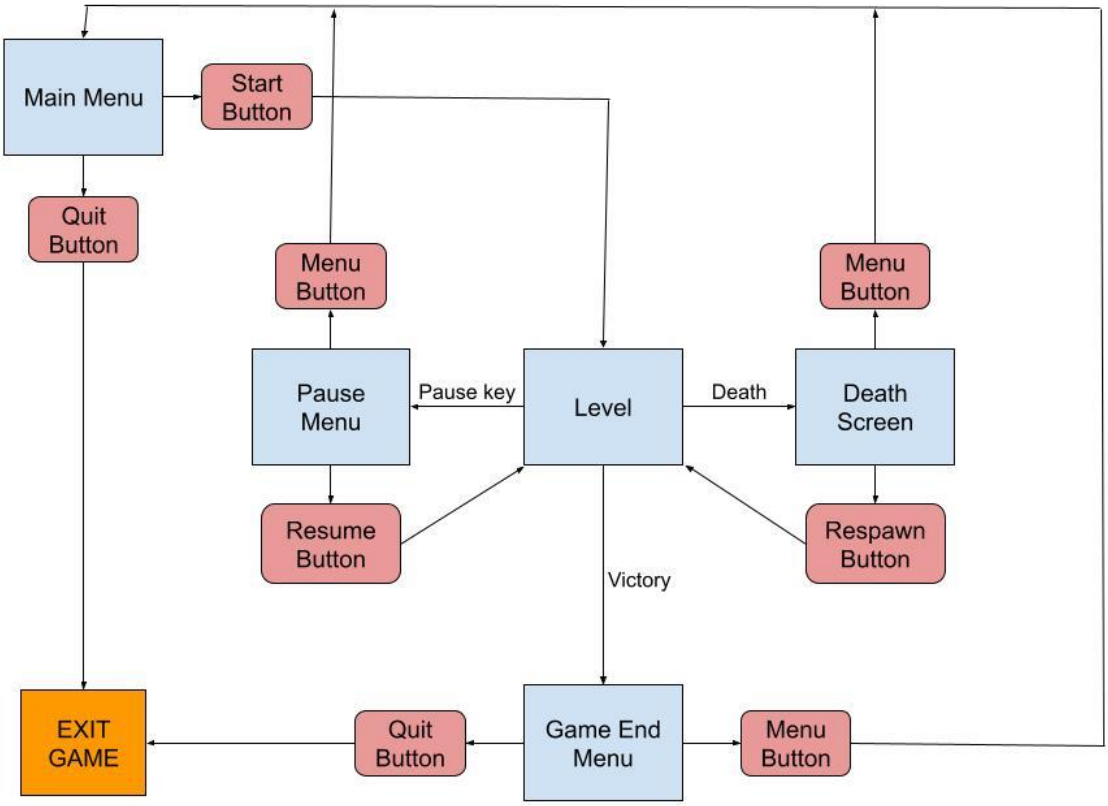
UI & Other Visuals

- A main menu, pause menu, death menu, and victory menu with functionality described in the "Screens" section and visualized in the Game Flow Diagram.
- Once the player has collected a jump powerup, an indicator will appear in the corner of the screen that tells them how many midair jumps they have remaining.
- The player, blocks, enemies, spikes, upgrades, and the victory orb will be composed of simple primitive shapes.
- The player, a simple green cube, will acquire animated orange wings once they collect their first jump powerup.
- Cars will have a death animation in which their sprites fly in all directions and rapidly fade. This will be accompanied by a particle effect meant to represent cascading oil.
- Spikes will have a slightly different colour if they are set to fall, to indicate their behaviour to the player.
- The jump powerup and victory orb will have a simple particle effect to indicate their importance.

Screens

- Main Menu
    - Start Game button
    - Quit button
- Victory Menu
    - Victory message
    - Play Again button
    - Main Menu button
- Death Menu
    - Defeat message
    - Respawn button
    - Main Menu button
- Gameplay Scene
    - Midair jump indicator

Game Flow Diagram



Non-Goals

Shadow enemy

Originally, the intent was to have a shadow-like enemy that follows the player and mimics their inputs. However, the complexity of such an enemy is outside the scope of the project.

Tools

Unity 2020.3.17f1

The Unity engine provides all the tools to quickly create assets for a 2D game prototype.

Visual Studio 2019

Visual Studio 2019 will be used because it comes integrated with Unity and supports C#, our scripting language of choice.

DOTween

DOTween will be used for simple visual enhancements. It is ideal for a small prototype since it requires no art assets and minimal effort while being quite effective for conveying aesthetic components.

Systems of Note

Menu System

The MenuManager script manages all menus. The pause menu and death menu are in the same scene as the level. They are toggled by being set active or inactive. The main menu and victory game end menu are in separate scenes from the level. When the MenuManager needs to make one of these menus available, it loads the relevant scene.

System Design Diagram