Game Proposal: Pathfinder

CPSC 427 – Video Game Programming

Team: Pathfinders

Name	Student Number
Aaron Zhang	44427862
Allan Jiang	34838748
Brayden Shinkawa	37212750
Daniel Lee	75182329
Joshua Chew	95081204
Kevin Zhu	56987274

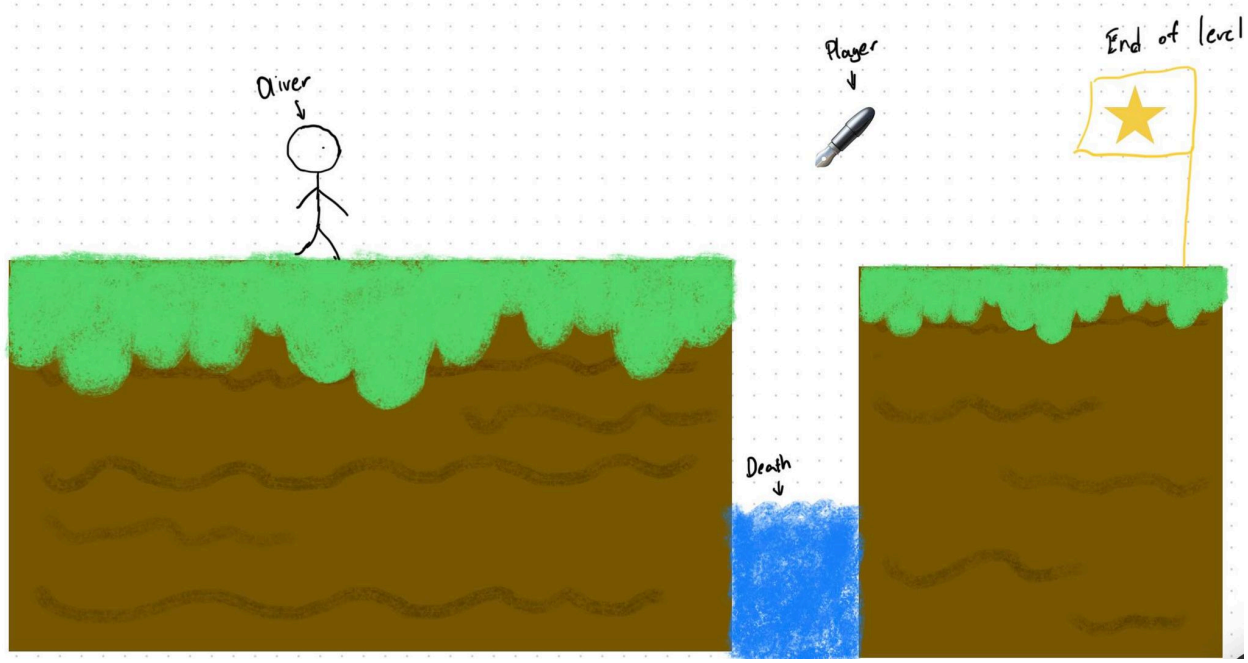
Story:

Our story is focused on the young boy Oliver who likes to draw his imaginations in his notebook. One night, he encounters a strange man who gave him a mysterious pen that transported him into a mystical world of his imaginations. The worlds are filled with different themes and levels, but amidst the fantasy-like environments, the worlds are also filled with dangerous traps and enemies. Your job is to help Oliver navigate through the terrain by using the magical pen. Use your imagination to draw him a path to safety using any means you like. However, be careful not to over use your powers as the magic pen only has limited ink.

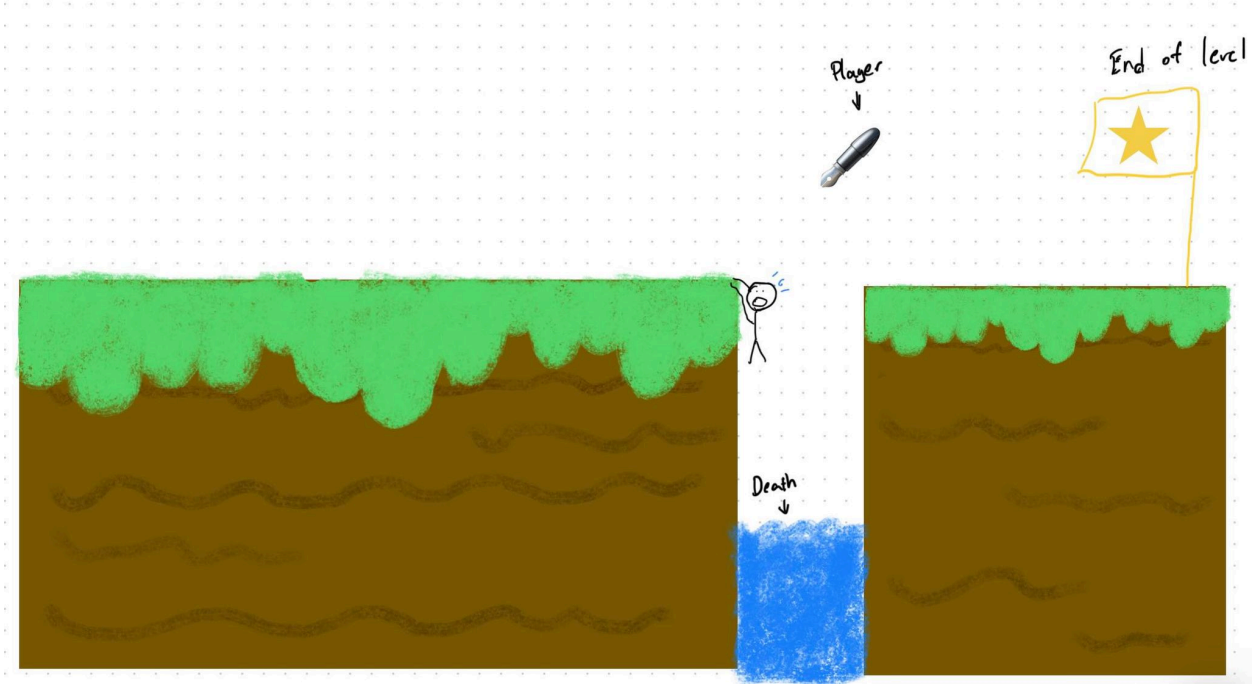
Traverse through the puzzle game in a 2d platform style with different mechanics in each level. Help Oliver combat the hot climate in a desert sand level, or make sure Oliver doesn't fall off the map in a slippery ice tundra. Regardless of the solutions you choose for each puzzle, your end goal is to help Oliver finish all the levels in the world, and escape back to reality.

Scenes:

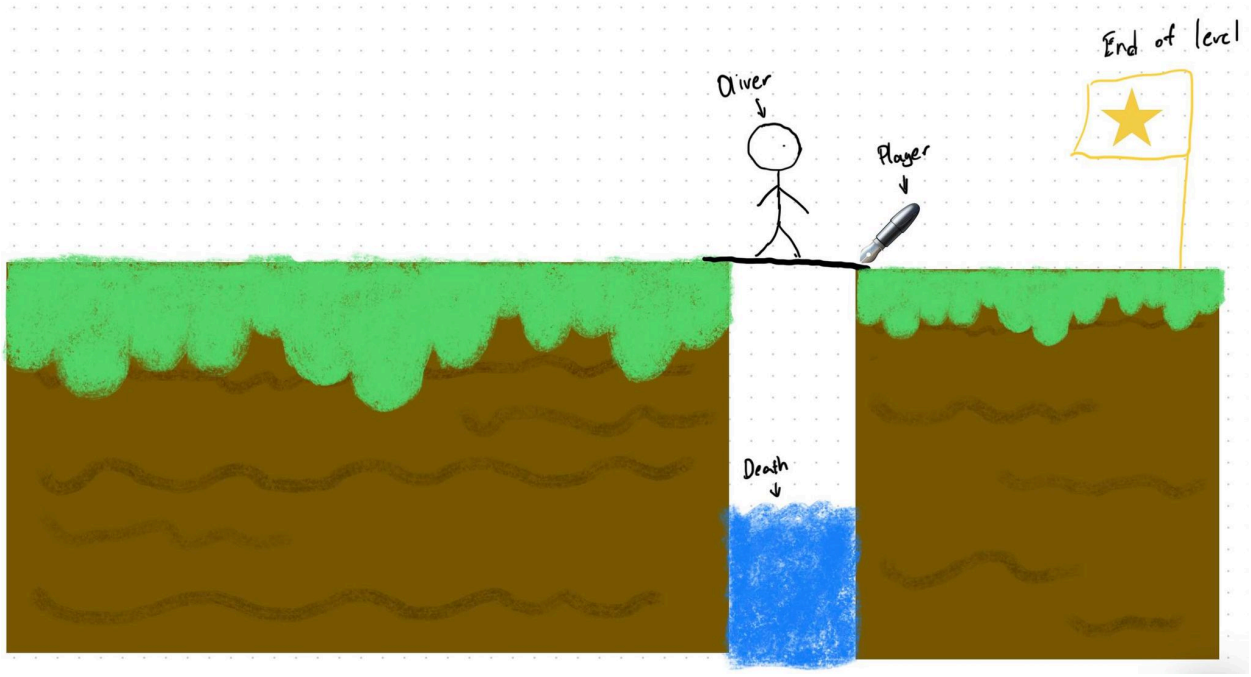
A rough design of the game's main screen, the player can go directly to play, check out the controls, or exit.



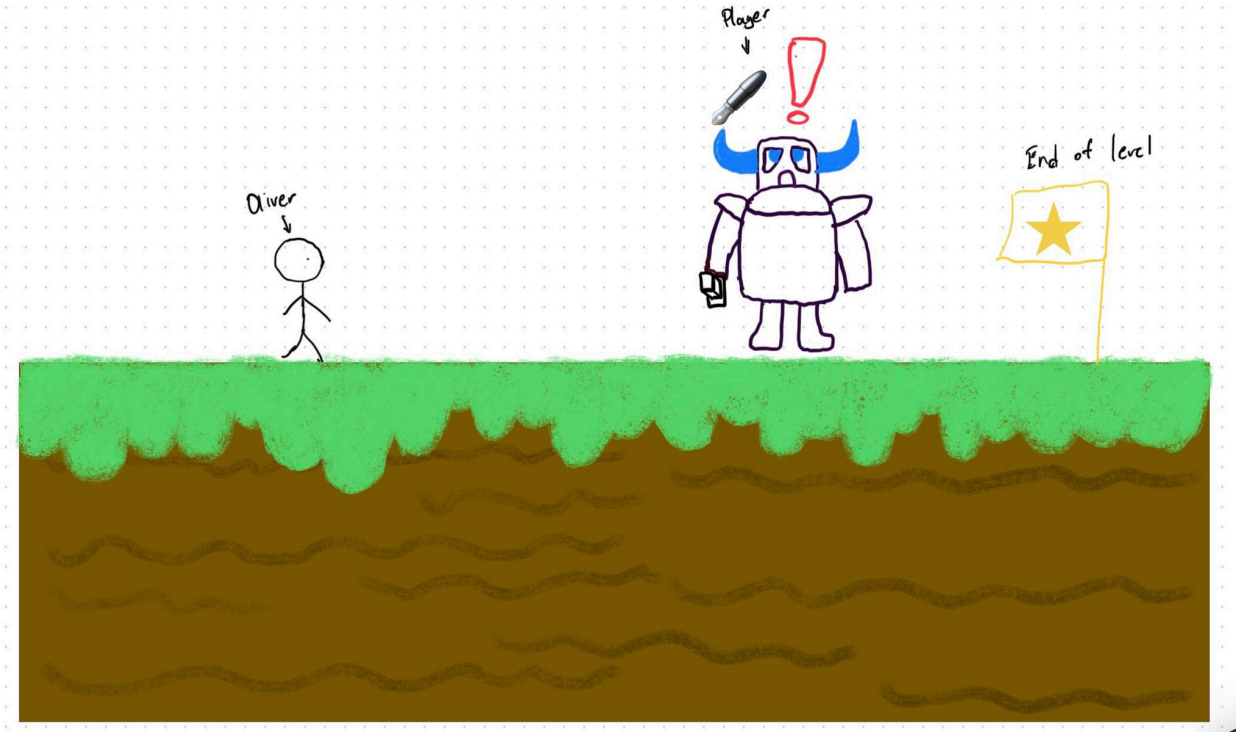
This scene shows the initial generation of a sample level. The player can control Oliver by moving him around with WASD keys, and the main goal here is to go to the next level by reaching the yellow star flag. However, Oliver is limited physically, and needs the player's help to get him through the stage.



In this instance, the player chose to do nothing with the pen, and tried to make Oliver run across the platform.



In this playthrough, the player drew a simple bridge with the pen before controlling Oliver to walk across the pit. Successfully allowing Oliver to step on the player generated ink shape, and progress to the next stage.



Finally, this last scene shows the enemy AI that has spotted Oliver through pathfinding, and the player needs to think fast before the enemy catches Oliver. The player can essentially do whatever they like with the pen to stop the enemy AI before the ink runs out.

Technical Elements:

Rendering

Our game leverages dynamic rendering techniques to bring players' drawings to life. Utilizing real-time rendering, the game transforms sketches into interactive paths and objects within the game world, allowing the player controlled stickman character to navigate through each level. This dynamic approach will enhance player immersion, making each gameplay experience uniquely engaging.

Assets & Sound

Our game will incorporate a library of assets, including a wide array of stickman animations, obstacle designs, and textural elements, to enrich the visual experience. From simple stick figures to complex environmental obstacles, each asset will support the game's diverse level designs and thematic requirements. This diversity in assets ensures that each challenge the player experiences feels fresh. Sound design serves a practical purpose by providing players with crucial feedback on their available resources. As players use the drawing pen to overcome obstacles, the volume or pitch of the accompanying sound effects dynamically reflect the amount of remaining pen ink. In addition to resource feedback, the game's sound design enhances the overall experience by introducing specific tunes for key events, such as hitting an enemy, passing or failing a stage.

Gameplay Loop & Logic

The game loop centers around a system that allows the player to (resourcefully!) draw shapes onto the scene which meaningfully affect the game world as physical platforms or obstacles for entities within the world. Then the player can control Oliver to use these newly drawn platforms to pass the level. As such, we can consider the game logic to be comprised of three core elements:

1. Movement of Oliver (and enemies)
2. Terrain manipulation and collision detection
3. Ink management

Players will be faced with new challenges each level; *the final level count is TBD, but we anticipate there will be between 5-10 levels available on release*, with a triumphant endscreen waiting for the players clever enough to complete all of them.

To make sure the game is repeatable, the puzzles for each level can be solved in numerous different ways.

2D Geometry Manipulation

Our game will allow players to create geometric shapes in real time. These shapes can serve as platforms, barriers, or tools to interact with the game world. The manipulation system is designed to support interactions between obstacles and the player, including precise collisions, thereby enabling players' creativity in overcoming challenges.

Physics

The physics engine will be designed to simulate realistic movements, weights, and collision effects. By incorporating a physics engine capable of handling various materials and their interactions, the game will offer a more challenging experience. Players will notice the different behaviors of various objects and surfaces, from the slide of ice to the bounce of rubber, adding depth to gameplay strategies.

AI

The enemy AI will be designed to pose a dynamic challenge to players, creating a gameplay environment that is engaging and unpredictable. Enemies will have set behaviors, such as patrolling specific areas or attacking when the player comes into a certain range. This level of AI complexity ensures that players must strategize to overcome obstacles and progress through each level.

Advanced Technical Elements:

Potential future scopes (some ideas..):
1. Power-ups: have different coloured ink that can have different abilities: yellow ink could be floating ink that doesn't fall, red ink could be hot ink that enemies and players cannot touch without getting hurt, etc. If we can't complete this, we can create levels where the power ups aren't needed, and instead the levels have to be solved in a more unique way without the extra abilities.

2. More complex AI: Introducing a wider range of enemy behaviors. Adaptive AIs with dynamic pathfinding that are able to interact with player-created objects and pursue the player. If we skip this feature, the AI may be more easily outsmarted by the player. If we can't implement a more complex enemy AI, we can instead add different enemies with different movesets, like bow and arrow enemies

3. Procedural content generation: Alternative would be more robust level design to not be so repetitive etc.

Devices:

We will use a mouse and keyboard in a windows environment. Left clicking and holding will draw physics lines. When the drawing is done, the player can click "ok" using their left-click mouse and can then walk using 'WASD' and 'space bar' to jump

Tools:

OpenAL will be used for a sound engine
SoLoud: Alternative audio engine, if we end up not using or needing the 3D capabilities.

Team Management:

Identify how you will assign and track tasks and describe the internal deadlines and policies you will use to meet the goals of each milestone.

We will meet up every single week on Thursdays and Saturdays to update one another on how the work is going, and if there is a problem ask to pair program with someone before the first week is done. Then we can have a buffer time with the second week to finalize any ideas.

We will also use OpenProject to plan out our development process, and GitHub pull requests to organize our incremental changes.

Development Plan:

Provide a list of tasks that your team will work on for each of the weekly deadlines. Account for some testing time and potential delays, as well as describing alternative options (plan B). Include all the major features you plan on implementing (no code).

Milestone 1: Skeletal Game

Week 1

- Set up basic shell for rendering game
    - Textured geometry, Basic 2D transformations, Key-frame/state interpolation
- Add controls for player character
    - Keyboard/mouse control
    - Coded action (i.e. drawing)
- Add gameplay elements
    - Well-defined game-space boundaries
    - Simple collision detection & resolution (between terrain and player)

Week 2

- Maintain Stability
    - Stable framerate + minimal lag
    - No crashes, glitches, or unpredictable behaviour
- Creative game element feature: Precise collisions [10] (advanced)
- Test plan
    - Test player movement all works
    - Make sure collision detection works between sprites

    - Test falling objects that are a “line” entity (for use in the drawing engine)

Milestone 2: Minimal Playability

Week 1

- Creating and linking Assets
    - Draw in-house or source externally (with credit and appropriate licensing)
    - Ensure they render on screen properly
- Focus on creating one single level
    - Construct terrain, obstacles, and enemy entities
    - Implement environmental hazards
- Program enemy AI
    - Implement movement path/circuit
    - *Stretch goal:* track player movement

Week 2

- Playtesting
    - Follow test plan, expand as necessary
    - Integration testing; assume components are tested individually first
- Debugging
    - Iterate over design, but do not change existing features (minimize error surface)

Milestone 3: Playability

Week 1

- Focus on creating next few levels
    - Design variety of environments
    - New hazards and physics challenges
        - Add other physics materials like ice, rubber, etc
- Add game menus
    - Buttons, toggles/switches, sliders
    - Provide UI to navigate level selection and enter/exit main game loop
- Implement sound design
    - Find or create sound assets, credit wherever necessary
    - Import and call OpenAL APIs
- Tweak component values for enjoyability of gameplay/QoL

Week 2

- Playtesting
    - Done per-feature added
    - Integration testing at the end
        - Add or tweak features according to player experience

- Debugging
    - After playtesting; lock in features for this milestone
    - Debug only according to above, adding no further features

Milestone 4: Final Game

Week 1

- Finalize all assets (sprites, animation, UI/UX elements, sound)
- Finalize level catalog and designs
- Create endscreen
    - Scrolling credits

Week 2

- Debugging
- Performance optimizations