Game Proposal: Bad Chilli Peppers

CPSC 427 – Video Game Programming

Team: Group 9

William Chow (93966943)

Suha Mansuri (31684327)

Sophia Zhou (55661094)

Simrit Nijjar (66234287)

Abhi Chouhan (81039513)

Story

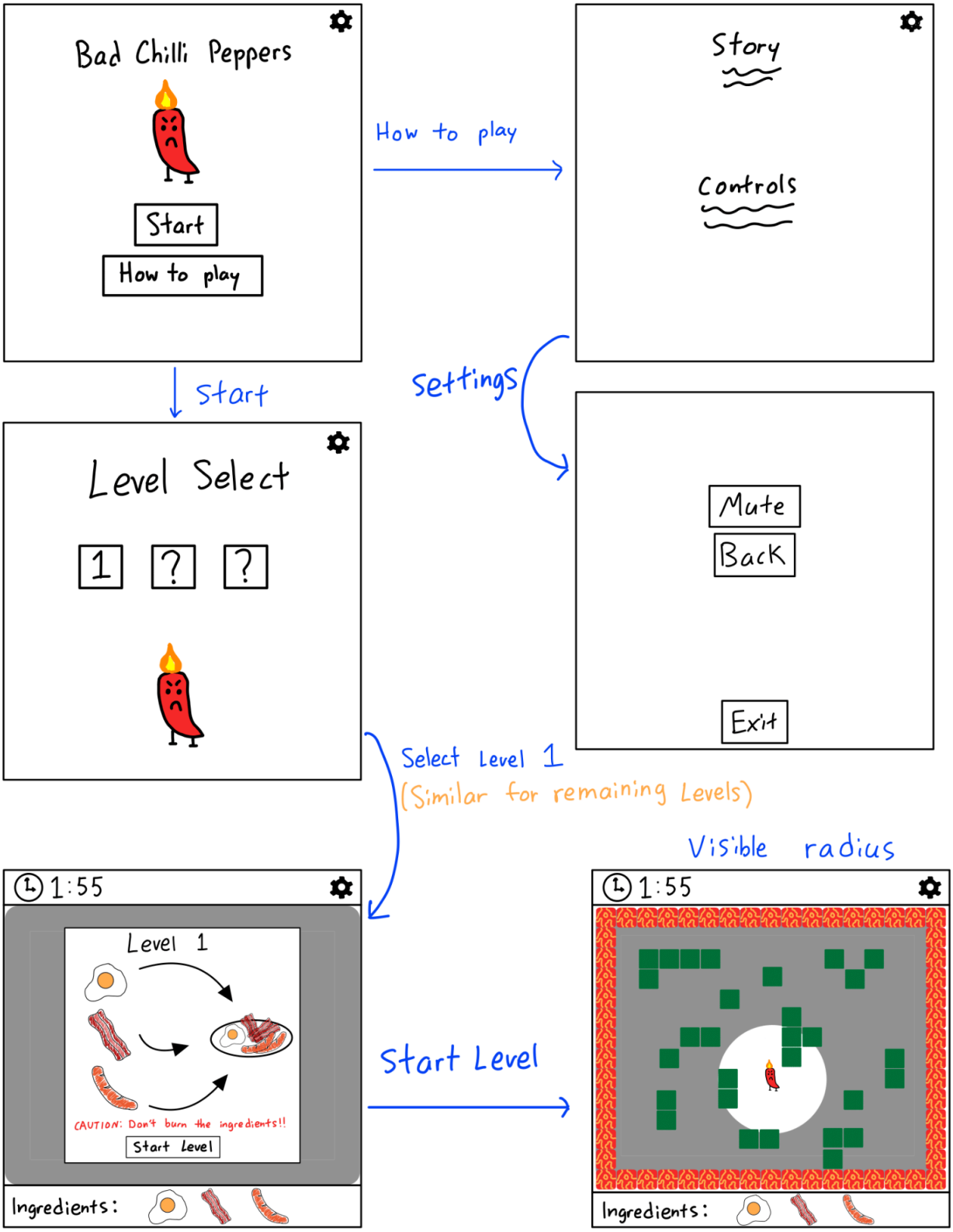
In Bad Chilli Peppers, you play as a chilli pepper on a mission to impress your beloved pepper partner with a series of extravagant meals. To achieve this, you must navigate through increasingly challenging levels, collecting the correct ingredients while avoiding obstacles, incorrect ingredients, enemies, and the time limit.

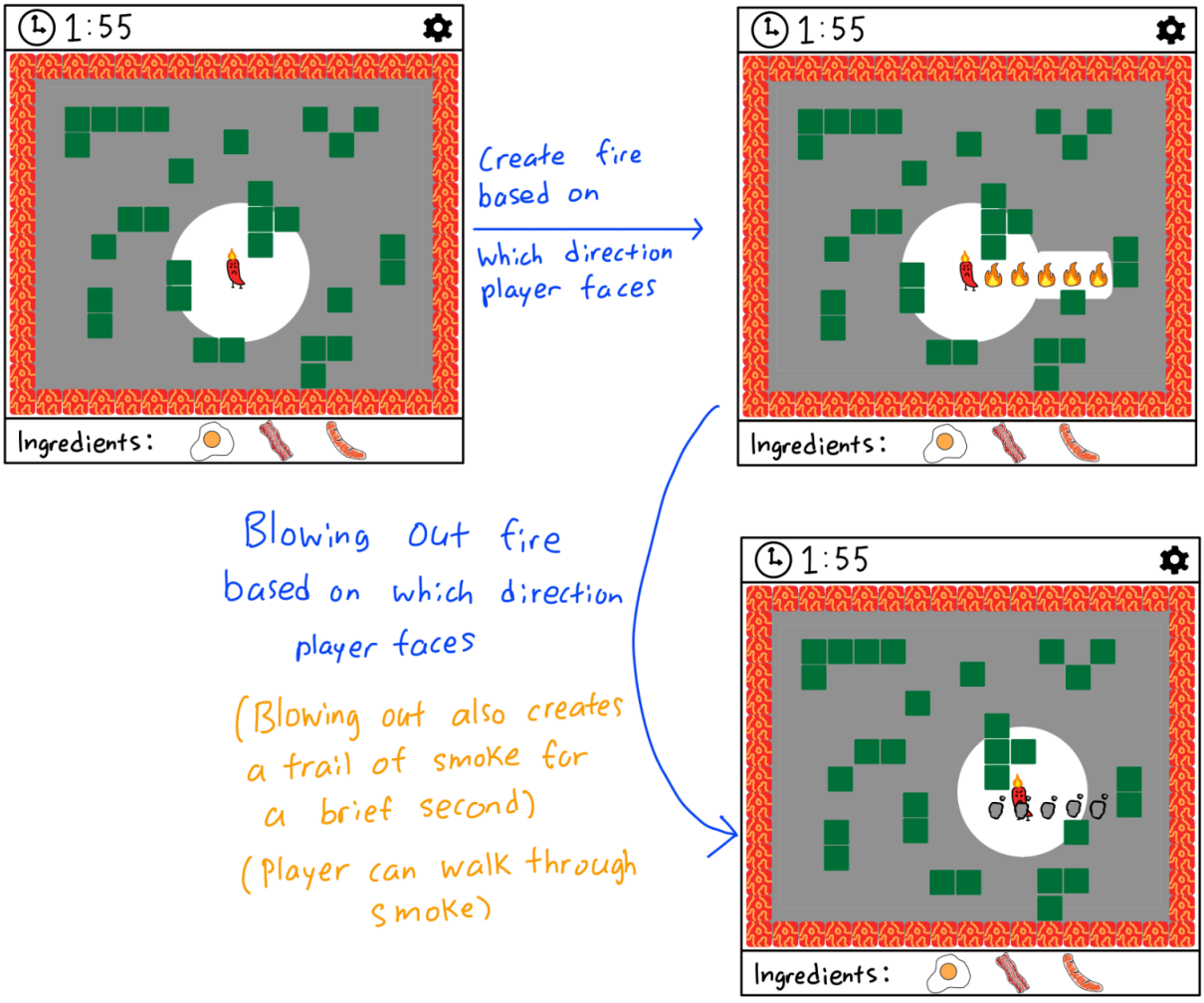
Armed with fiery breath, you can create and destroy walls of fire to illuminate dark areas, block enemies from attacking, and strategically navigate the level. However, you must also be cautious– your flames can also burn the ingredients you need, forcing you to wait for them to respawn and adding pressure to the clock. Your goal is to collect only the required ingredients, as collecting incorrect ingredients will deduct time as well. To pass the level, you must collect all the ingredients before the time runs out, while fending off a variety of enemies.

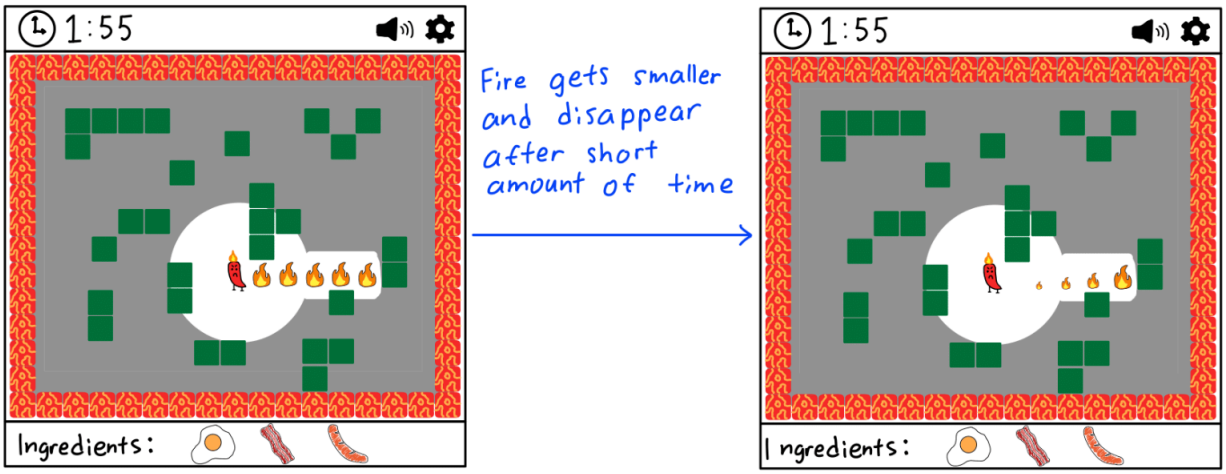
Enemies will be able to chase you, change directions, or create and extinguish fire making it difficult for you to pass each level. Visibility also plays a key challenge in the game. While you can see the entire map, enemies, ingredients and power-ups remain hidden until they come within your visible range. Fire blocks will be visible at all times. To aid in your progress, power-ups will be scattered through the levels that can temporarily boost your speed or increase your visibility range. However, the enemy will also be able to pick up the power-up which can either speed them up or increase the distance and rate they can extinguish fire.

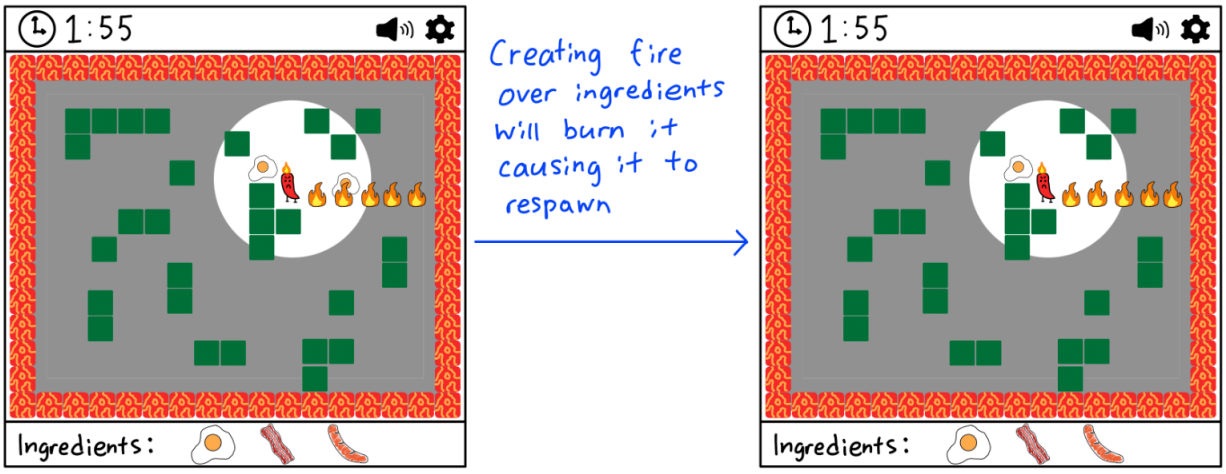
Once you have completed all the levels, the story ends while the chilli pepper cooks the extravagant meal, and the chilli and their pepper partner live happily ever after.

Scenes

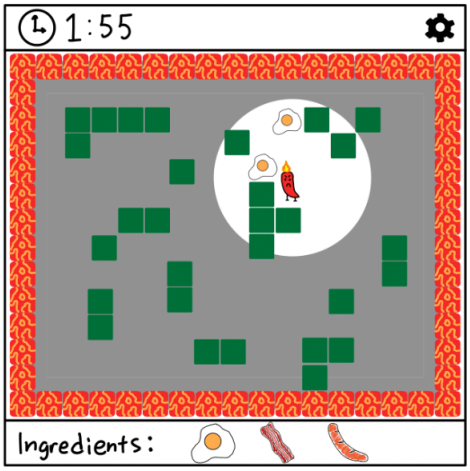

Bad Chilli Peppers
Start
How to play
How to play
Story
Controls
Start
Settings
Level Select
1
?
?
Mute
Back
Exit
Select Level 1
(Similar for remaining Levels)
Visible radius
1:55
Level 1
CAUTION: Don't burn the ingredients!!
Start Level
Ingredients:
Start Level
1:55
Ingredients:


1:55
Ingredients:
Create fire based on
which direction player faces
1:55
Ingredients:
Blowing out fire based on which direction player faces
(Blowing out also creates a trail of smoke for a brief second)
(Player can walk through smoke)
1:55
Ingredients:

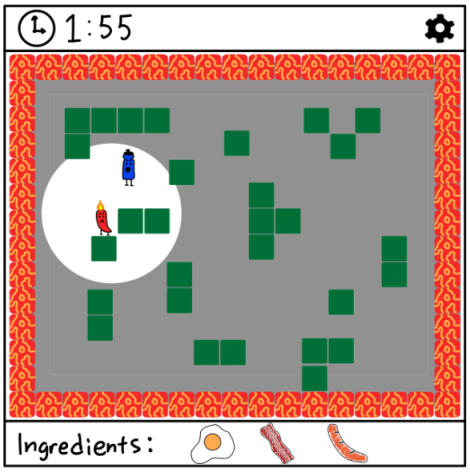

1:55
Ingredients:
Fire gets smaller and disappear after short amount of time
1:55
Ingredients:


1:55
Ingredients:
Creating fire over ingredients will burn it causing it to respawn
1:55
Ingredients:

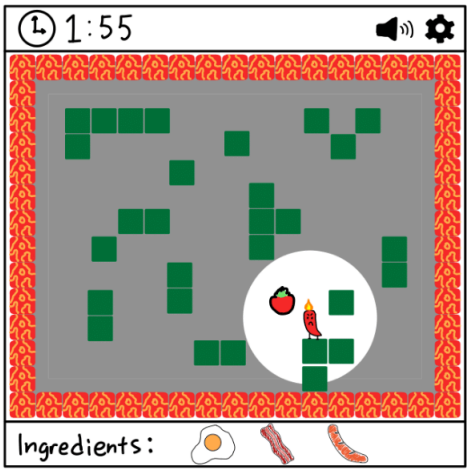
Ingredient visible within radius



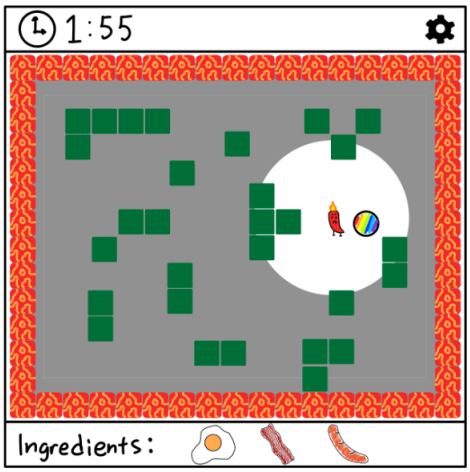
Enemy visible within radius



Incorrect ingredient visible within radius

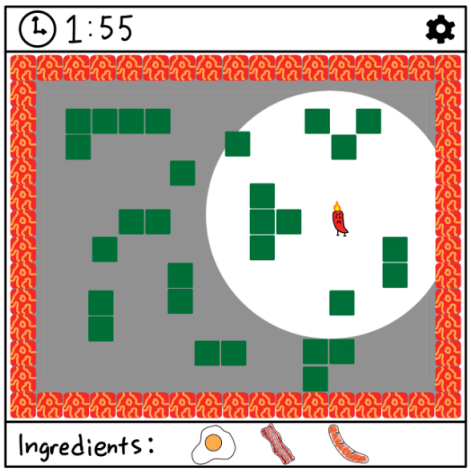


Increased visibility powerup visible within range

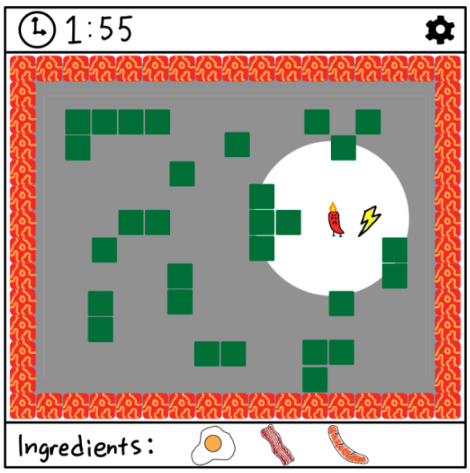


Consuming powerup

Consuming powerup grants increased visibile radius

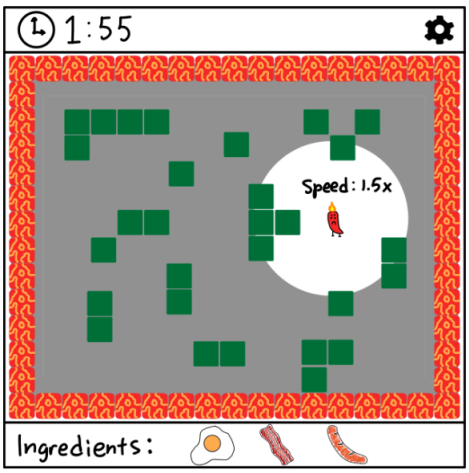


Increased movement Speed powerup visible within range

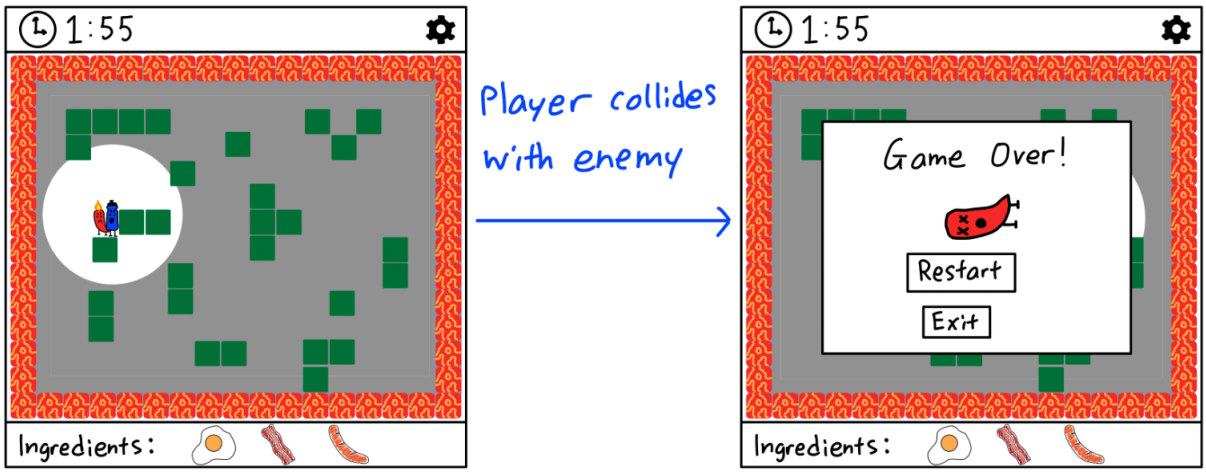


Consuming powerup

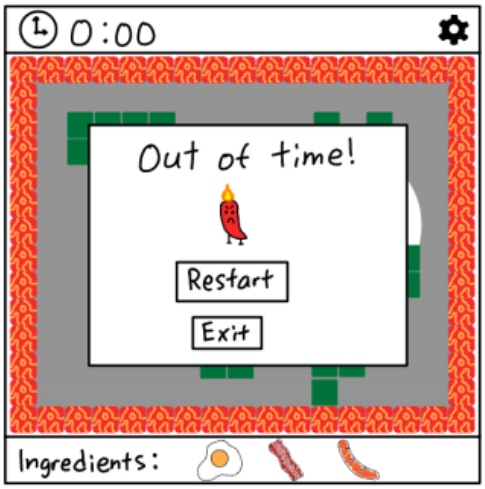
Consuming powerup grants increased movement speed



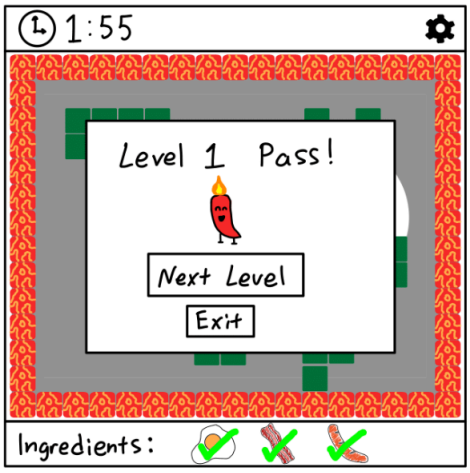
Death Screen



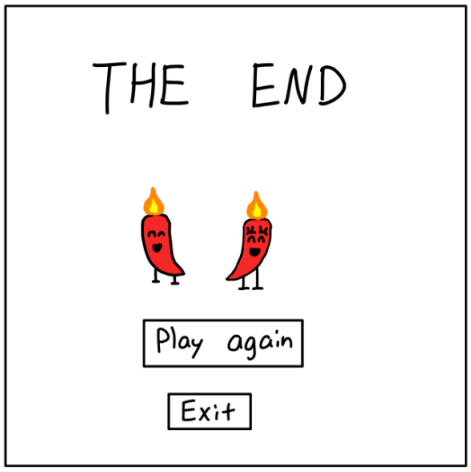
Time runs out



Level Pass



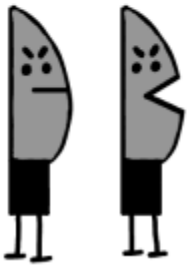
Victory Screen after passing all levels



Enemy Types

Water Bottle

Knife



Torch

Technical Elements:

Rendering

Background tiles and static objects such as walls, lanterns, and ingredient blocks will be rendered first with each position set by the coordinates on the 2D grid.

Dynamic entities such as the player, enemies, fire blocks, and certain ingredients that can move will be rendered on top of static elements.

- **Player**
    - The chilli pepper will be rendered to walk around the map as well as create and destroy fire using the flame on its head
- **Enemy**
    - The enemies will be rendered for basic movement.
    - The torch-type enemy will be rendered to both blowout player-created fires and create fires to destroy ingredients.
    - The knife-type enemy will use an AI path-finding algorithm to follow the chilli-pepper, recalculating every ‘x’ amount of frames.
    - The water-bottle-type enemy will simply move around the grid, bouncing off walls, obstacles, and fire blocks.
- **Fire Blocks**
    - The player-created fire blocks will be rendered dynamically when the player presses the spacebar to create. Blocks will be created in a straight line in the direction the chilli is facing, generated block by block. There will be a half-second delay between the creation of each block. Each fire block will have a timer, that will gradually decrease the fire in size until 0, then generating a smoke block in its replacement
- **Power-ups**
    - These will be spawned randomly, with a timer and position that will change with each level.
    - A sound effect would be played when the power-up spawns, and another sound effect if the enemy consumes the power-up, notifying the player.
    - The screen would show the general direction of the powerup to the player (subject to change based on game playability/difficulty)
- **Smoke Blocks**
    - Smoke blocks will be rendered dynamically when the fire blocks extinguish. They will have a timer, get smaller in size as the timer heads to 0, and then disappear.
- **Movable Ingredients**
    - Movable ingredients will simply bounce off walls around the screen, with no timer and a dynamic position (subject to change based on playability/difficulty)

Sprites

- Chilli pepper player sprite
    - One for the player
    - One for the pepper partner
- Enemy sprites of three types:

    - Torch-type enemy (creates and extinguishes fire)
    - Chaser-type enemy (knife)
    - Standard enemy (water bottle)
- Ingredient sprites
- Fire blocks
- Powerup sprites
    - Rainbow for increased visibility (player) or increased fire creation abilities (enemy)
    - Lightning for increased speed for both player and enemy
- Background tiles
- Lantern blocks
- Wall blocks
- Larger obstacles
    - Volcano
    - Lava pools

UI Components

- Timer
- Settings
- Pause
- Main menu
- Mute button
- Volume slider

Geometry

- Player, Ingredients, powerups, blocks and enemy geometries will have positions represented as squares aligned to the grid map.
- Their geometric properties such as (x, y) position, width and height will determine interactions with other entities.
- The player and enemies will be slightly taller than a grid cell to add depth to the rendering, with the hitboxes remaining the size of a cell.
- Ingredients are confined to the size of a cell, along with powerups and blocks.

2D Geometry Manipulation

Collision detection will be essential for our game, taking place in four forms:

- **Player and Obstacle**
    - Collisions with obstacles such as walls, lanterns, or fire blocks will prevent our player from moving past them
- **Player and Enemy**
    - This will trigger the level-over condition when the player collides with an enemy and trigger the level-over screen
- **Player and Ingredient**
    - Correct ingredient: This collision will progress the player toward completion of the level. Reduce the count of ingredients left to collect.

    - Incorrect ingredient: This will take away an estimate of 10 seconds for every player-ingredient collision
- **Enemy and Obstacle**
    - Enemies cannot pass through obstacles including fire blocks that the player can create, incentivizing players to trap enemies in blocks.
    - The torch enemy will have fire creating and extinguishing features that can do this block by block instead of a full line. If the torch enemy has consumed the rainbow powerup, it can create and extinguish full lines of fire.
- **Enemy and Ingredient**
    - The Torch enemy type will be able to light ingredients on fire
    - Any other enemy type will just move through the ingredient
    - Enemies will be able to collect power ups

Gameplay logic/AI

The player can move up, down, right and left using the WASD keys to collect the correct ingredients and avoid the enemies and bad ingredients. The player will be able to create/destroy fire using the spacebar. The fire will last for an estimate of 15 seconds. There will be logic handling the key detections for these interactions.

The player can interact with the game environment in different ways:

- When the player reaches a wall, an obstacle, or a lantern they cannot pass through it. They must move in a different direction.
- The main goal of the chilli pepper is to collect all the correct ingredients by navigating past obstacles, enemies, and incorrect ingredients. A collision detection system will be used to register when the character touches the corresponding entities.
- The player can use its fire to burn incorrect ingredients, trap enemies and light up the map. The fire creating ability will have a cool down of an estimate of 3 seconds to prevent overusing the ability.
- The player can blow out fire to create a new path it can walk through.

AI will be used in the form of pathfinding algorithms where the Knife-type enemies will use the A* algorithm to find the most optimal path to chase the player. The enemy will constantly try and find the shortest path to the player. As the player moves the AI will periodically update the pathfinding to ensure that the enemy is consistently following the player.

Implementation

- Maintain a 2 way data structure allowing for quick lookups for which entity is on each grid position
    - Would need to maintain a Grid → Entity along with the regular component → Entity
    - Subject to change as implementation continues

Physics

The physics aspect of our game will include fire smoke particle effects that are inversely impacted by gravity, with some random motion. Additionally, smoke blocks will also have particle effects that are again inversely impacted by gravity with some random motion.

Advanced Technical Elements:

- **Dynamic Lighting**
  - The map will be dark besides the areas lit by the player's flame and other light-emitting objects
  - Omitting will allow the player to see the entirety of its current view
  - The alternative would be to only allow the player's flame to light the area for easier rendering
- **Camera Panning for Position-based View of the Map**
  - The map will follow the player (assuming the map is large enough)
  - Omitting will force a more rigid transition during movements
  - The alternative would be to omit the position-based view of the map entirely and allow the player to see the entire map
- **Fractal Rendered Map**
  - After creating the structure for rendering the levels, create an algorithm to auto generate the levels by filling the structure and ensuring there is not enclosed spaces
  - Omitting will reduce the number of levels available in the game
  - The alternative is to stick to hand-designed maps and limit the number of levels to 3
- **Shadows**
  - Objects will block lighting
  - Will account for multiple lighting sources to render shadows
  - Omitting will allow the player to view entities behind walls as long as they are within the visible radius
  - The alternative is to allow this as a default functionality

Devices:

- **WASD Keyboard** → Simple WASD + Action key layout for movement and ability control
- **ARROW Keyboard** → Arrow keys + Space bar layout for a *potential* second player

Tools:

- **GLFW** → Windowing
- **SDL2** → Audio playback
- **Tiled** → Rendering tile-based levels

Team management:

- We will have a weekly meeting, as of now said for Sundays, where we will discuss the following:
    - Development plan for the week
    - Any supplemental meetings for the week that are necessary
    - Discuss progress made from the previous week
    - Discuss the following week's goals.
- If there are any issues, these will be discussed in the supplemental meetings where members can reach out for help from other group members if needed, or if someone is running behind on a deadline.
- Our policy is that you must be present during the weekly meeting and the supplemental if you are needed, and actively respond to ensure everyone is on track each week.
- We plan to keep track of tasks using the agile approach
    - We will also maintain a strict code review process for each PR to ensure quality is maintained
    - Will assign two reviewers while requiring at least one to approve
    - PR author must add tests to their changes that must pass against the PR reviewer

Development Plan:

Milestone 1: Skeletal Game

Week 1

- ☐ Set up project repo
- ☐ Implement basic game window with GLFW
- ☐ Render a basic tile-based level (We can use Tiled library)
- ☐ Implement player movement using WASD
- ☐ Load and display a player sprite
- ☐ Basic collision detection → boundary of map

Week 2

- ☐ Implement enemy sprite with basic AI for random movement
- ☐ Create start screen and other basic UIs
- ☐ Load ingredient sprites
- ☐ Test for movement and collision detection → ingredient and enemy

Milestone 2: Minimal Playability

Week 1

- ☐ Add ingredient collection logic with a count feature
- ☐ Add power-ups
- ☐ Add timer countdown
- ☐ Introduce enemy types in different levels

- [ ] Add fire creation and destroying
- [ ] Sound effects

Week 2

- [ ] Add visibility mechanics
- [ ] Add death and success frames and collision detection queuing death frame/success frame
- [ ] Ingredient recipe screens etc
- [ ] Testing

Milestone 3: Playability

Week 1

- [ ] Add multiple levels and unlocking enemies
- [ ] Add game menu features (pause, retry, restart, main menu)
- [ ] Finish story frames for start and finish

Week 2

- [ ] Save and load game state?
- [ ] Full play game and testing

Milestone 4: Final Game

Week 1

- [ ] Bug fixes
- [ ] Final effects

Week 2

- [ ] Final audio effects
- [ ] Bug fixes
- [ ] Final visual effects
- [ ] Extensive play testing