Game Proposal: Party Time

Team Doge

Team Members

Name	Student Number	GitHub Handle
Heng Wu	31696198	pushen
Johann Cooper	24173312	johannc
Luis Bolanos	71762264	luisb
Mac Iverson	40125163	macman1
Wenkai Cai	90787698	wenkai02
Yiyi Yan	40800161	yanyiyi

Story

The overall game will be structured in rounds. The first half of the round involves all 4 players (turn based) to move along the main board through a random dice roll. After all players have gone, a mini-game will launch (either free for all, or teams depending on the game). The winner of the mini-game will receive a coin prize. This repeats until all rounds are exhausted, at which point the winner is declared based on total winnings.

The game is based on Mario Party. It is a very intuitive game and does not require particularly hard skills which makes it fun for new players and kids. The mini-games will allow us to incorporate technical skills required by the assignment without it being completely out of place on the main story line. We are going to a similar theme as the Mario Party.

Technical Elements

- **Gameplay logic:**: The game will feature character sprites with animations for movement/death etc.
    - Random number generators will be used for dice rolls to control player movement on the board.
    - The game will keep track of what state it is in (ie. player's turn, round number, etc.), and save the player's progress at checkpoints.
    - The game will continue until there is a winner in the main game. Mini-game will end whenever a winner has been declared.
    - Include the multiple player component for the main board, and AI system in mini-games for players to compete with.

    - There will be around 6 mini-games
- **AI:** AI will be really simple on the main board (ie. random chance of using an item, triggering a random dice roll). On mini-games AI will be more complex which may include path finding, or auto targeting at the player.
- **Geometric/Sprite/Other assets:** We will create all the textures for characters and background, as well as the animations.
- **2D Geometry manipulation**: Some animations will use keyframe animation on the 2D geometry.
- **Rendering**: 2D game with 3D assets.
- **Physics**: Some mini-games will feature momentum & gravity
- **Audio:** The game will feature audio feedback for movement, dice rolls, menu scrolling etc.

Advanced Technical Elements

- Smooth camera animation that travels between the 'current' player on the main board
    - Backup: static camera high-up seeing entire board OR camera teleports to next player without smooth interpolation
- Self-learning AI players that will get better as the game goes on, or pre-trained to have different difficulty levels, up to almost impossible to beat.
    - AI players will have some kind of path finding algorithm, auto targeting methods and reinforcement behaviors to be competent enough, making the human players feel pressured.
- Complex physics or simulations
    - Having environments in mini-games that require some kind of customized physics engine for simulation.
    - Incorporate different physics into gameplay itself
- Rewarding yet challenging gameplay design
    - The game should be difficult enough that the player has to do well in the mini-game to succeed, but not to a point where the player cannot come back after making some mistakes.

Devices

We are planning to use mostly a keyboard, and possibly with the assistance of a mouse, to control the character or select options. The game will be supported on MacOS/Linux/Windows, through Zoom's remote control feature.

Since it's a multiplayer game, we will take multiple inputs from different players. Since there is a consistent delay with the remote control, and only one player can control the game, we will rotate each player controlling the game. For example, in the mini-game, the first player, player 1, will complete the game, and then the game will wait until player 2 takes the remote control, and etc.

There is also the possibility of using the zoom API to take input from zoom chat, and control the game from there. However, more research is needed to check the feasibility of this method of control, and the result will still be quite limited, since rapid movements are difficult to control with only zoom chat.

Detail list of controls

Keyboard:

WASD*: control the player movements, and select menu items*

ARROW keys*: alternative for WASD*

R*: roll the dice SPACE BAR*

S*: go to the game setting menu. It will pause the game.*

ESC/Q*: quit the game*

P*: pause the game*

MOUSE*:*

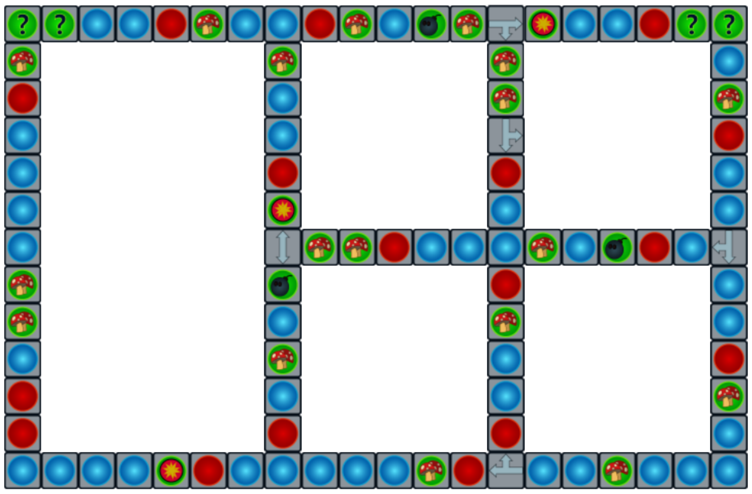
Click dice to roll

Click setting to go to setting menu

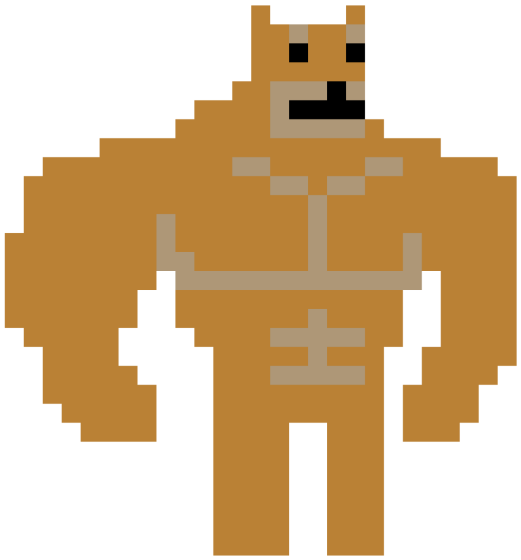
- *Include resume, restart and quit*
- *Include game settings, like difficulty*
- *Include display settings, like resolutions*

Concepts

2D Board layout:



Sample Main board layout, power ups and different options


Sample player character, the Doge

Tools

AI

- Path planning algorithm
- Trajectory planning
- Deep Q-learning to train AI players

Rendering

- Look at different libraries, just in case there is something better than the standard ones
- Libraries that can help with multiple player input sources
- OpenAL for Audio playing library

Math

- Eigen, Mathfu for computation libraries
- Random Terrain Generation

Game Design

- Quest for basic game story design
- Piskel for animated sprites and pixel art generation

- Tiny Game Design Tool for mini-game design
- Photoshop for assets design and animation

Zoom API

- We will need to integrate with Zoom's chat feature to get player inputs
- Use Zoom API and Get data from API urls

Development Plan

Skeletal Game

Week Sep 27:

- Set up code structure, workflow and project management tools
- Complete the system design - System partially complete (scene management is currently a palace holder)
- Main Board proof of concept programming
- Mini-game proof of concept programming- moved to Week of Oct 4th.
- Basic textures for players and backgrounds.

Week Oct 4:

- Main Board (graph multiple directions). Graph system in place, have yet to add direction controls
- Mini-game starting code (decide who the winner is at random no actual mini-game)
- Basic animations and audio
- Brainstorm and design mini-games
- Test zoom API to see if we can capture player inputs - test results in 2 players (local and zoom controller) able to give input to the keyboard at the same time. We wanted more players so we decided on more turn-based mini-games

Minimal Playability

Week Oct 11:

- Design and create physics engine for mini-games, feasibility test ideas
- Finish scene management system, and ensure input is not handled on all scenes, just the active one.
- Add items to the game (ie. mushrooms) and the item system
- Add UI elements.

Week Oct 18:

- Finalize the mini-game design
- Add basic mini-games - split work between everyone

- Mini game 1: players can collect coins, avoid falling objects, can be controlled with keyboard or mouse gestures?
- Add the round system.
- Add board mini-game with physics of momentum and friction.

Week Oct 25:

- Mini game 2:
- Mini game 3
- Mini game 4
- Start testing the game loop
- Add directionality to the board.

Week Nov 1:

- Add polish with higher quality assets and animations
- Add camera focus on player feature
- Keep working on mini-games
- Integrate main board with mini-games
- Add basic AI into developed mini-games
- Mini game 5
- Mini game 6
- Mini game 7 (for backup)

Week Nov 8:

- Testing the game loop
- Augment the arena with more mechanics
- Keep adding polish with higher quality assets and animations
- Improve AI players in mini-games

Week Nov 15:

- Testing the game loop
- Decide what mini-games to keep and what to remove
- Add advanced features into the game

Final Game

Week Nov 22:

- Finalize the game features, stop adding anything and start testing
- Improve code quality, add documentations
- Remove extra components and clean up the game.
- Polish all the assets one last time.

Week Nov 29:

- Testing the final game

- Bug fixing and patching
- Prepare for a demo, finishing up documentation.