ZHENGMAO OUYANG for MR CERONES ICS4U ON DEC 12 2018

COMPUTER SCIENCE FINAL DETAILED PROPOSAL

The game I will be creating is an action platformer based off of the game “Bad Ice Cream” called “Dairy Free”, in which players, represented by differently flavoured ice cream icons, travel around each level collecting fruits while avoiding vegetable monsters. Players collect fruits by walking over them, though the fruits have varying properties. Some will actively avoid the players. Others will remain stationary. The vegetable monsters have varying properties as well, from blindly going in random directions to actively tracking the user. Bumping into an enemy means instant death for the user. Each level is grid based, and players have the ability to cast a row of ice blocks in the direction they’re facing. When facing a row of ice blocks, the user can destroy those rows as well. The level restarts when all players have died. Each level is complete when all fruits have been collected and at least one player remains standing. Over time, the levels increase in difficulty, and the monsters become more intelligent. Note that this differs from my entry for the General Topic Submission, which would not have been feasible within the given time frame.

Like the Sokoban game, levels will be generated using textfiles with symbols that represent the different tiles that will make up the level. Possible tiles include brick walls, ice floors, ice blocks, melons, metal hazard blocks, vortexes that transport you to another location on the level, heat blocks that melt ice cast on them after some time, or cold blocks that will regenerate ice on that tile after some time. The default tile is snowy ground.

The location of fruits will be generated using another textfile. Possible fruits include watermelon, grapes, lemons, oranges, strawberries, limes, tomatoes, chili peppers, and kiwi. Each fruit will be worth 100 points to the user that collects it. Kiwi will move in random directions. Strawberries will move away from the user. Tomatoes randomly change position every five seconds. Chili peppers move about randomly but melt surrounding ice blocks. When a fruit and ice block exist on the same tile, whether it’s by default or due to the user creating those blocks, the fruit will be stuck within the block. The ice block then must be broken before the fruit can be collected.

The location of monsters will be generated using a third textfile. Monsters include broccoli, spinach, eggplant, celery, brussels sprouts, onions, and ginger. Broccoli walks in a straight line until it reaches a wall or ice block, at which point it will find a new, random direction to walk in. Spinach may change direction at any time, though randomly. Eggplant is able to smash ice blocks while randomly changing direction. Celery moves randomly for five seconds at a time, and then chases the user for another five seconds. Brussels sprouts can break adjacent ice blocks on all sides while moving randomly. Onions always chase the player, but are unable to break blocks. Ginger burrows underneath the ground every 6 seconds and follows the player, reappearing after 3 seconds. A moving dot on the screen shows the path of the ginger.

Each level is time-limited. Beginning levels will start with a time limit of 180 seconds, moving up to 600 seconds. At each level’s completion the score is calculated using the amount of time left, as well as the fruits collected. If there are multiple players, each player will have a different score depending on how many fruits that player collected, and at the end of each level, the player with the highest score will be crowned the winner of that level.

Objects will be used for each interactable tile, each fruit, each monster, and each player, as well as the timers. Each object will handle its own location, state, and actions. This allows for a compartmentalized and structured organization of the program.

As for controls, players can either play together on one keyboard, or use Arduino controls. If time allows for it, users may also be able to use online rooms with a unique identifying keyword that allows for up to 3 players to play at a time over LAN. Regardless, the maximum number of players is still 3. If all on one keyboard, Player 1 will use the arrow keys to move and space to create or destroy rows of ice blocks. Players 2 and 3 can use WASD and F, IJKL and H. Arduino controllers will use 5 buttons, each connected to a different port. When the button is clicked, electricity will be allowed to flow to the port, and digital readings will show an ON state for that port. Though each level is grid-based, characters' movements will be continuous, rather than discrete.

Here's how some of the Required Program Components are met:

Mathematical operations will be used to place objects on screen in a regular fashion. Users may name their characters through an input. End-level screens showing points serves as an output. Arrays of similar objects will be used to handle game behaviour. A sorting algorithm will be used to sort users' scores to determine their order. String methods will be used to trim usernames' trailing and leading spaces. At the beginning of the game, the user will be met with a main menu to either start the game, read instructions, or see Credits. When starting a game, they will create a save file where they can go back to the level they were previously working on. The game will allow for many save files, so the user will be able to search for the name of theirs. Before any of the main menu is even shown, a password will be required to start the program.

Any other Required Components are self-explanatory within the context of this game. The game will also feature animation and sound, building on the static graphics explored in Sokoban.