Design Document

INTRO

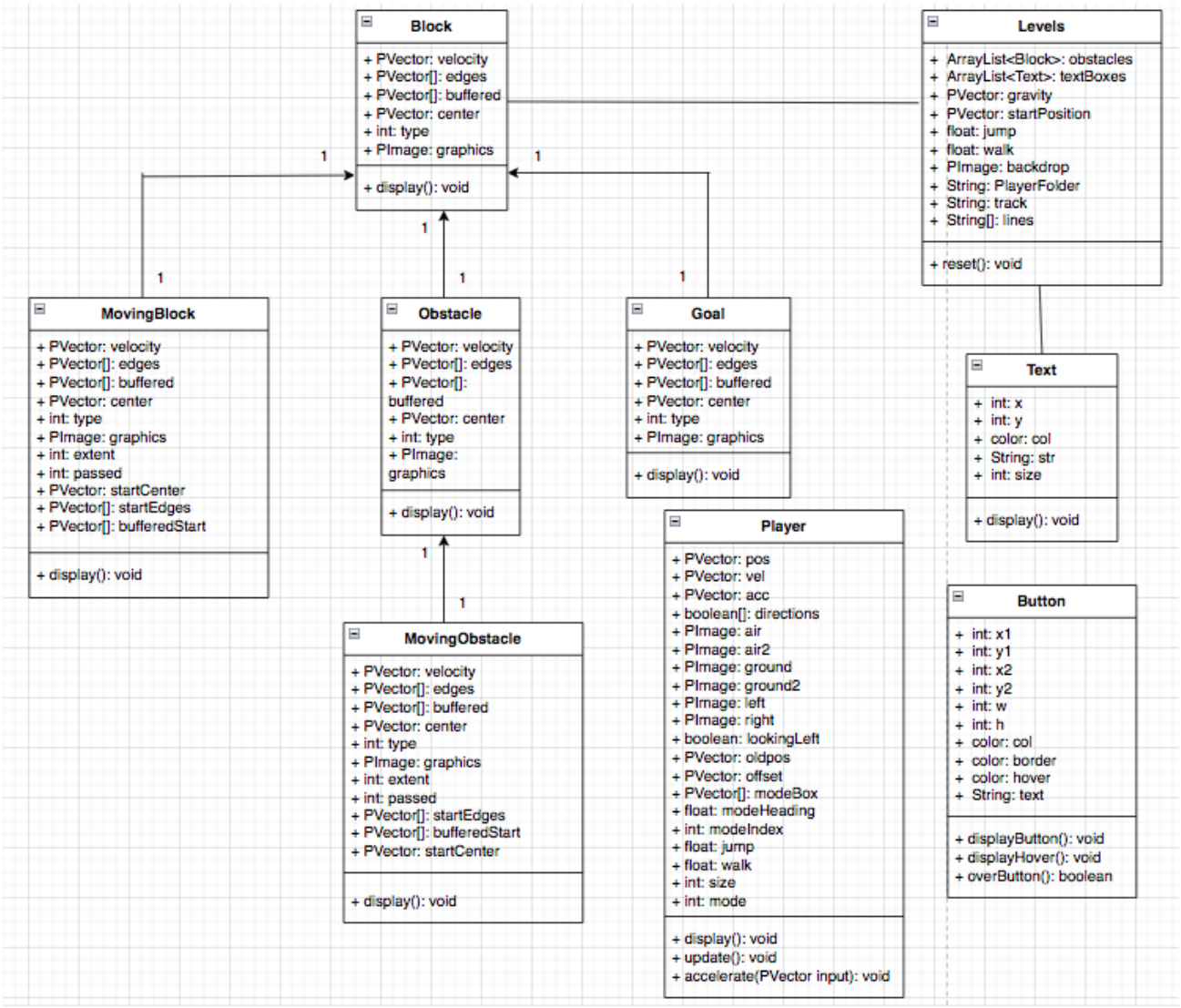
Period 6.
Rachel Uvaydov and Matvei Karp
Group Name: RunoSong
Project Title: Kantele Quest

DESCRIPTION

Our project is a 2D platformer game inspired by Finnish mythology (the Kalevala runosong) and the Finnish kantele instrument. It starts with a title screen that leads to the level select menu where the player can choose one of our 15 levels to play. There are multiple mechanics that get introduced and explained in the beginning of the game, and each new level tests the player's skills dodging obstacles, riding moving platforms, and jumping to go through doors and collect strings to build their kantele. To do this, we created a Player class for the character the user controls, with attributes that track its location and movement. Whenever an arrow key is pressed, the velocity changes, so the position of the character changes as well. There is also a Block class, encompassing platforms the player can stand on (Block and MovingBlock), obstacles (Obstacle and MovingObstacle) that 'kill' the player, and the goal that the player has to reach to complete the level (Goal). All of these are subclasses of Block, which tracks the location and edges of the objects so collision can be implemented. The user can reset a level at any time or pause the game, allowing them to go back to the level select screen or the title screen. Once the player completes the last level, they are taken to a win screen and can replay the game as many times as they'd like.
The levels in the game are implemented through .txt files (a template of which is included in our repo). There is a level class that reads through the files and creates an array of Blocks and an array Text based on the file for each level that get displayed when that level is being played.

Our game also implements sound and music with the Beads library. Each level .txt file specifies the location of the music to play for it. This is then interpreted within the main file, which creates a new SamplePlayer and adds it, along with a Gain, to the AudioContext of the game. Whenever the player is reset or moves to the next level, the SamplePlayer is set to be killed when it reaches the end and is then set to the end. Following this, a new Sample Player is created. The kantele music is pulled from a performance by Tuomi Orava (https://www.youtube.com/watch?v=rkUIGHmnw8E).

UML DIAGRAM



HOW DOES IT WORK?

The objective of this game is to complete all 15 levels, gathering strings every 5 levels to create a Kantele. The user controls a character using the arrow keys. The character walks when the user presses the right or left arrow keys, and jumps when the user presses the up arrow key. The longer the up arrow key is held down, the higher the jump of the character is. If the character is next to a wall, by walking into the wall and repeatedly pressing the up arrow key, the user can get the character to

climb up the wall. The user can reset the level they are playing at any time by pressing the 'R' key. Additionally, they can pause and unpause a level by pressing the 'P' key. In the pause menu, there is a button that leads to the level select screen, where the user can select a level to play, or go back to the title screen. Upon completing the game, the user is taken to a win screen, which has a button to take them back to the title screen so they can replay the game.

FUNCTIONALITIES/ISSUES

Week 1.

Currently, we have tentative designs of the Player and Block classes, each utilizing different methods of implementing the classes, as well as templates for the main KanteleQuest file. We've implemented one method of creating the blocks and player that utilizes a grid of squares on the screen, and another one that instead has a freely moving character that checks for collisions by checking for valid intersections between the line segment that the character is moving and the line segments that make up the edges of blocks. Both have methods of creating and displaying the blocks and player, but we're having issues with implementing movement of the player and correct collision with the blocks, so the player can't clip through the blocks. By next week, we are planning to be done with implementing the Player and Block classes, such that the player can move with correct velocity and acceleration and won't clip into the blocks. Then, we'll work on creating the Obstacle class, as well as implementing moving blocks and obstacles.

Week 2.

This week, we had a lot of trouble fixing the issues with the player movement and collision with blocks, but this issue was fixed by reworking the collision "beam" to come from each corner rather than from the center. Once the issue was solved, the Block and Obstacle classes were able to be implemented, as well as a Goal class that serves as the goal for each level. Due to these issues, we weren't able to create MovingBlocks and MovingObstacles as we originally planned, and instead started working on features needed for the pause menu and level select screen of the game. The first step for this was creating a Button class, creating buttons whose appearance can be changed by hovering over them, and that can change screens in the game by clicking on them. The game can now be paused by pressing the "P" key, which pulls up a pause menu with a button that will give the player the option to go to the level select screen (which does not exist yet). Additionally we implemented a system for storing level data as .txt files and created a class that reads those files and converts them into the variables needed for the main file. Since we were not able to implement moving

blocks and obstacles this week, we would like to be done with that by next week. Additionally, we will create a working level select screen with buttons that lead to different levels, and design the layout of each level so we can implement the blocks and obstacles. We will likely also try to add sounds, settings and configurations that the player can change, and a title screen for the game by next meeting.

Week 3.

This was the last week to work on our project, and we were able to finish creating our game. Sadly, we were not able to add everything we originally intended on putting in the game. We never ended up adding a settings menu, as we couldn't actually think of anything to put in the menu. Also, we had been thinking about adding mechanics that change the physics of the game, such as ice physics or wind, but we had a lot of errors with collisions in our Block class and its subclasses, so we were never able to focus on adding something else. This is also the reason we never added enemies (besides just stationary/moving obstacles) for the player to defeat, as everything in the game was tied so heavily to the Block class that creating a different class for an enemy rather than just creating blocks is something we never got around to doing.

However, we did add a lot of other features. We completed the moving blocks and moving obstacle classes, added music, and added multiple UI features. There is now a title screen at the beginning of the game with a button that leads to the completed level select screen, with pathways to every level. Additionally, we finally designed all the levels in the game, 15 completed levels, with updated player sprites matching the Finnish mythology theme, updated images for the Blocks (brick and crate textures), Obstacles (a spiked object and falling hammers), and Goals (doorways or kantele strings), as well as updated backgrounds for all the levels, a plain black screen as opposed to 3 temporary photoshop backgrounds made a couple of weeks ago. There are also now edges to the screen (invisible Obstacles), so the player cannot fall off the map. We also added a Text class, to display text introducing and explaining concepts to the player at certain coordinates during different levels, and implemented the Text in the Levels class and .txt files. Lastly, we added a win screen that pops up after the last level is completed, with a button that takes the player back to the title screen so they can play the game again.

There were multiple issues we encountered while doing this, especially when it came to designing the levels in the .txt files. There was an issue with the blocks not loading correctly in levels, despite their coordinates being correct, which we realized was because the coordinates had to be in a certain order for the collisions with the blocks to work properly. There were also some issues with adding the Text class to the

.txt files, as they were meant for the Block class and subclasses, but they were fixed with a lot of reading over the code and usage of if and else statements. Our game is still not perfect, there are still some collisions issues and the key inputs are sometimes slow, but we were finally able to complete it, combining all of our efforts in the past few weeks to complete this game.

LOG

Rachel:

- Constructors and display() methods for Player and Block classes with grid method
- KanteleQuest file that creates and displays a grid of squares with SQUARE_SIZE, which can draw blocks and player on specific grid squares
- Button class that displays a button with text (to be used in the pause menu, level select screen, and title screen), with functionalities to display the button differently when the mouse hovers over it, as well as changing the window when the button is pressed
- Added pause menu to game that can be activated and deactivated by pressing "P"
- Added Button ArrayLists to KanteleQuest, one for buttons that stay on screen, and one for pause menu buttons, allowing different buttons to be pressed during different states
- Edited designs for and implemented many levels
- Created Text class to display text at specific coordinates, and implemented the class in the Levels class and .txt files
- Created title screen and win screen, and updated colors for buttons, text, and the pause and level select menus

Matvei:

- Constructors, display() methods for both Player and Block classes with the "collision beam" method
- Player.update(), Player.characterMotion(), KanteleQuest.keyPressed(), and KanteleQuest.moveCharacter(), all involved in checking and moving the character in accordance with key presses
- KanteleQuest file that creates a character, establishes gravity, and sets up obstacles.
- Fixed the above to actually work(removed characterMotion(), fully reworked moveCharacter() and keyPressed())

- Shifted all level-dependent information into txt files that are read by Levels objects, preparing for a level select system.
- Added a set of blocks {Block, Obstacle, Goal} that are now defined by a series of points, enabling triangles and the like.
- Added the collision effects for those block types
- Implemented block movement and made players be pushed by blocks that move
- Continued designing the level txt format
- Implemented limits to movement.
- Replaced all rect() with images
- Implemented background audio
- Designed a few levels
- Designed textures for blocks
- Found backgrounds and music