Brief Description

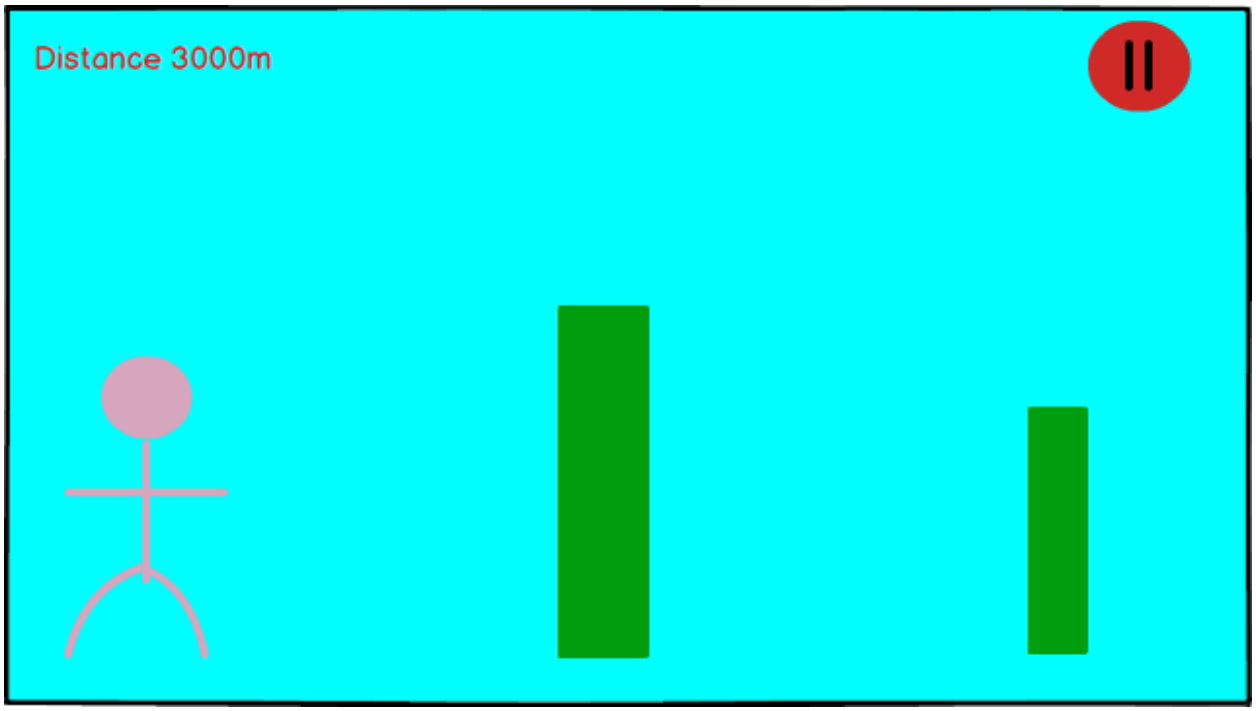
Our group plans on creating an infinite running sidescroller game in the same vein as popular mobile app and flash games such as Copter and Flappy Bird. Our main character will be Jadrian and as the user progresses through the stage of the game, the difficulty of the game will steadily increase. We will randomly generate obstacles and enemies on the stage for the user to avoid, and as the user progresses through the game, the obstacles will get closer together and speed up how fast the user 'runs' through the game. If the user hits an obstacle the game ends.

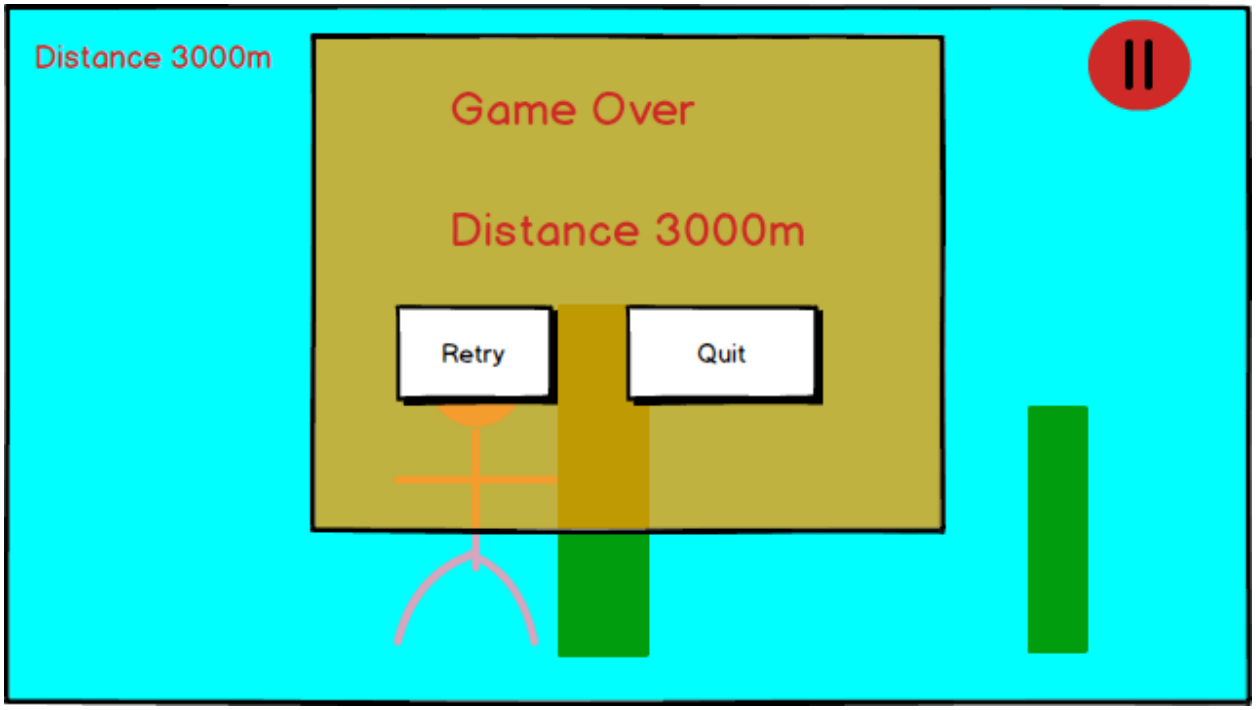
Feature List

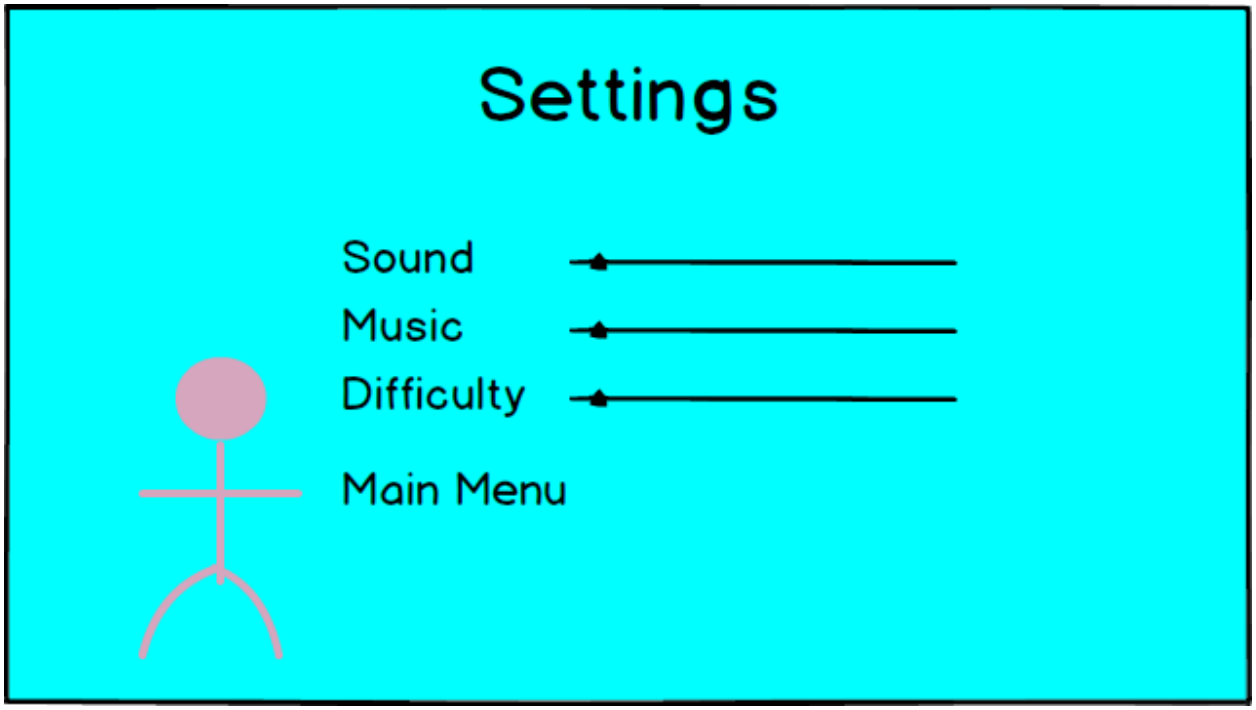
- Randomly generated continuous stage
- Stage is populated by different obstacles/enemies which can KO user
- Animated pixel sprite for main character
- Some degree of control over character movement (horizontal movement determined by model, speed at which level scrolls, but user can make the main character jump)
- Score counter to record distance progressed before KO
- High score board which allows user to record score (later stages of development)
- More detailed/animated stage backgrounds, possibly include different regions/themes as character progresses further in the game (later stages of development)
- More detailed animation for character movement, and enemies/obstacles (later stages of development)
- More sophisticated character movement, possibly duck/slide or attack obstacles and enemies (later stages of development)

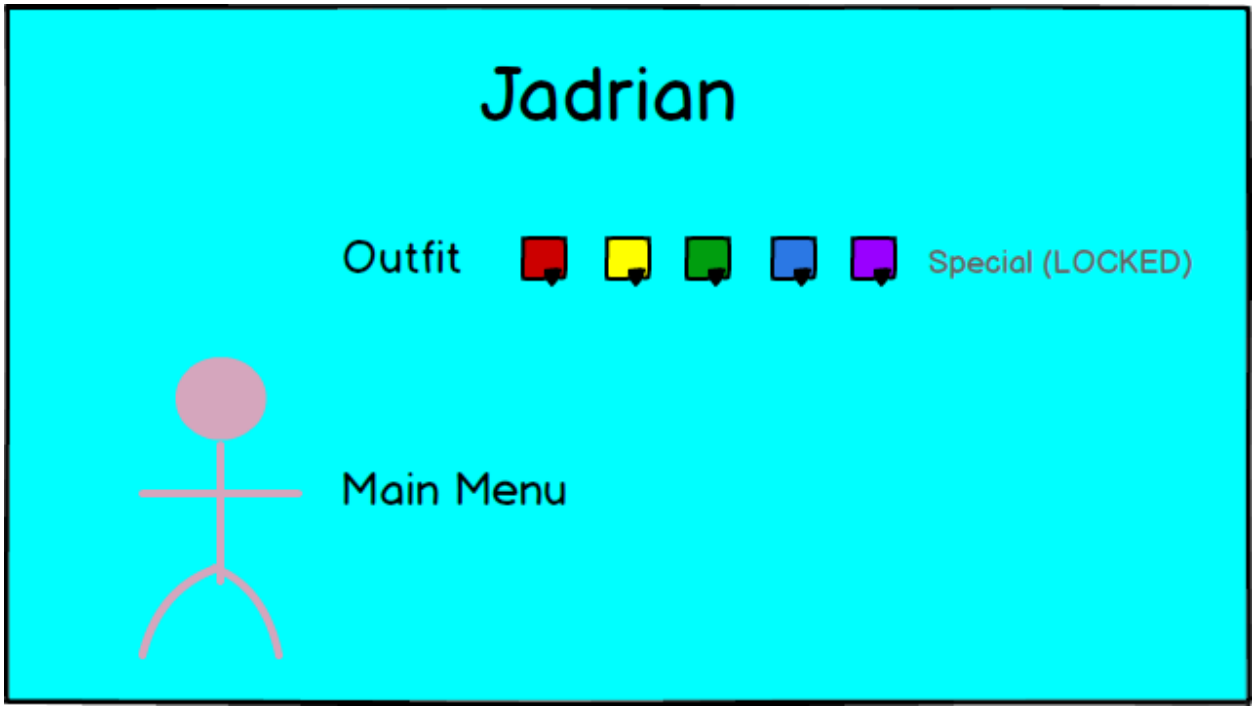
Mockups


Jadrian Runner
Levels
Jadrian
Achievements
Settings
Quit

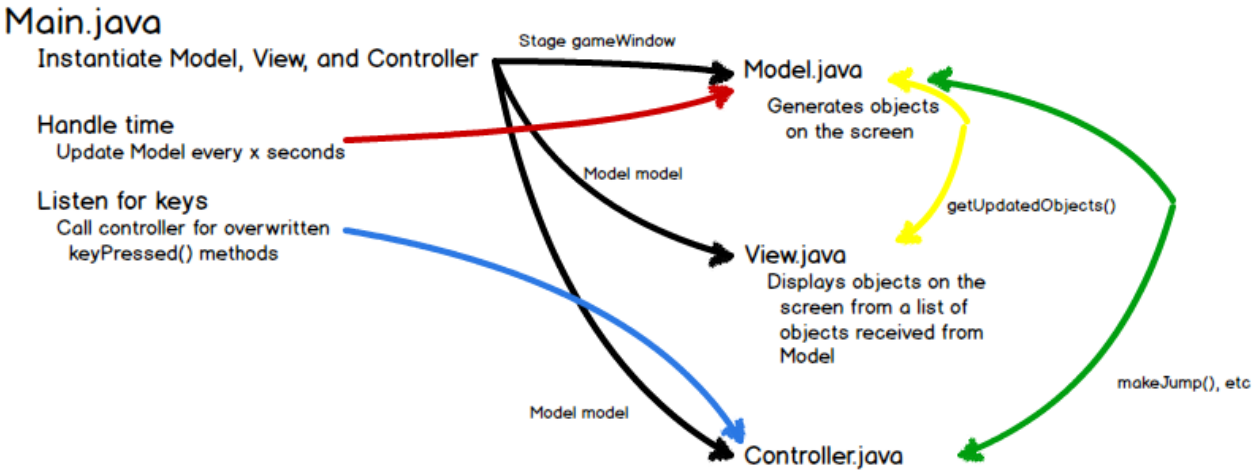

Distance 3000m


Distance 3000m
Game Over
Distance 3000m
Retry
Quit


Settings
Sound
Music
Difficulty
Main Menu


Jadrian
Outfit
Special (LOCKED)
Main Menu

System Architecture



Signatures of Major Software Components

- **Main.java [class]**
  - *Main class for our game stores variables for javaFX, and the game timer which model uses. Very bare-bones other than that.*
  - void start(Stage gameWindow)
    - *Sets up the stage, and sets the scene using loadMainScene().*
  - void main(String[] args)
    - *Calls launch method, passes args.*

- **Controller.java [class]**
  - Controller(Model model) [constructor]
    - *Creates instance variables to store state of user input. Takes reference to game model as argument for the purpose of calling methods based on user input.*
  - void updateModel() [method]
    - *Calls corresponding method of model to respond to user input when user input has occurred.*
  - void resetInputState() [method]
    - *Resets instance variable for state of user input after model has been updated.*

- **Model.java [class]**
  - *Class that handles the generation and movement of objects for display on the screen. Also keeps track of the score.*
  - Model(Stage gameWindow) [constructor]

        - *Instantiates the*
    - void updatePositions(int dTime) [method]
        - *Updates the positions of all of the sprites on the screen, given the change in time*
        - void moveBackground(dTime)
            - *Updates the positions of each of the background sprites*
        - void movePlayer(dTime)
            - *Updates the positions of the player sprite*
    - void updateJump(int dTime)
        - *Wrapper for canJump( ) and makeJump( )*
        - void makeJump(int dTime)
            - *Changes the player's velocity*
        - boolean canJump( )
            - *Helper method for updateJump. Determines whether or not it's valid for the player to jump at this time*
    - void updateTrack(int distance)
        - *If needNewTrack( ), calls genNewTrack to generate the new track. The type of obstacles and how many of them appear depends on the current distance.*
        - boolean needNewTrack( )
            - *Returns true if we're far enough that we need a new track*
        - void genNewTrack( )
            - *Generates the next screen over the previous screen (leaving the current screen intact)*
    - void startGame( )
        - *Generates the initial positions of the various positions*
    - void quitGame( )
        - *Exits everything*
    - void loadMainScene( )
        - *Resets the scene for the Stage gameWindow to the main menu layout, and sets event handling for elements of the scene.*
    - void loadGameScene( )
        - *Resets the scene for the Stage gameWindow to the gameplay layout, and sets event handling for user input in controlling the gameplay.*
    - void loadOptionsScene( )
        - *Resets the scene for the Stage gameWindow to the options menu layout, and sets event handling for elements of the scene.*
    - List<Sprites> getUpdatedObjects( )
        - *Returns the updated objects for the View to display on the screen*

- **View.java [class]**
    - *Displays the scene based upon the information Model has interpreted from the user input.*
    - View(Canvas canvas, Model model) [constructor]
        - *Takes the canvas and reference to the model as an argument, and stores a reference to the canvas and its graphical context as instance variables.*
        - void drawGameScreen()
            - *Calls the model and gets the game objects to be drawn. Calls the appropriate methods to draw each of the elements.*
        - void drawRunner(GameObject runner)
            - *Draws the runner sprite based on current state of object passed as argument.*
        - void drawObstacle(GameObject obstacle)
            - *Draws the obstacle sprite based on current state of object passed as argument.*
        - void drawGameOverScreen()
            - *Calls the model and gets the objects to be drawn. Calls the appropriate methods to draw each of the elements.*

Test classes for unit tests

- **ModelTest.java [unit test class]**
    - testCanJump( )
        - *Tests if canJump( ) returns the correct boolean value when the player sprite is on the ground and when the player sprite is in the air.*
    - testMakeJump( )
        - *Tests if makeJump( ) correctly adjusts the velocity of the player object*
    - testUpdateJump( )
        - *Tests if updateJump( ) correctly interacts with the two above methods given a valid time for a jump and an invalid time for a jump*
    - testMoveBackground( )
        - *Tests if moveBackground( ) shifts the positions of each object correctly for a few different changes in time, including zero and a negative time.*
    - testMovePlayer( )
        - *Tests if movePlayer( ) shifts the positions of the player object correctly for a few different changes in time, including zero and a negative time.*
    - testGenNewTrack( )
        - *Tests if genNewTrack( ) creates a new track of the correct size, and within the correct range of obstacles*
    - testNeedNewTrack()

        - *Tests if needNewTrack( ) returns the correct boolean for whether we need a new section of track for a position before, at, and after the position we decide it should do so at.*
    - testStartGame()
        - *Makes sure that we startGame returns the correct types of objects.*

- **ControllerTest.java [test class]**
    - General unit tests:
        - *Test constructor, make sure it builds the right type.*
        - *Test update, make sure it calls the correct method in Model.*
        - *Test reset, make sure it resets for all kinds of input.*