System Request

Project Name:
Bowman in Java.

Team Bon Java Members:

Noah Beilke	Austin FitzGerald	Tim Collier
Software Engineering	Software Engineering	Software Engineering
beilken@uwplatt.edu	fitzgeralaus@uwplatt.edu	colliert@uwplatt.edu

Project Sponsor:
Doug Selent, PhD
Assistant Professor of Computer Science and Software Engineering
selentd@uwplatt.edu

Business Need:
Bowman is a game where players take turns firing arrows at each other, with a random and unknown (to players) distance between each player. Health damage is based on where an arrow hits the player model. Players can use a velocity vector to get their shots closer to the other player. There is no up-to-date version of Bowman, and no version created on any platform besides Adobe Flash - which is deprecated.

Business Requirements:
The platforms and software that are required for this project include: Slack, Eclipse IDE, Jira, Visual Paradigm, Java 10.0.x, JavaFX, Photoshop, Scene Builder, and the e(fx)clipse plugin. Subversion will be routed to a remote repository provided by Dr. Selent.

Business Value:
The goal is to create a platform-independent version in Java, and develop new features that no Bowman game has ever included. Users of the game will enjoy improvements in software quality, such as support and mechanics. If we were to do an economic feasibility analysis, our return on investment would be undefined; our cost is non existent. Creating an updated version would be an educational experience for our members as well as increase the player base for Bowman games by at least three people.

Special Needs:
The Bowman in Java game will require creation or adaptation of sprites used for object models, they will be created in Photoshop.

Software Requirements Specifications

Bon Java

Austin FitzGerald, Tim Collier, Noah Beilke

Functional

- **Input**
  - o *FN-IN-01 The system should allow a user to select the difficulty.*
  - o *FN-IN-02: The system should allow a player to choose a name and a color.*
  - o *FN-IN-03: The system should allow all text characters for player name choice.*
  - ~~o *FN-IN-04: The system should allow a user to choose the texture beneath their character*~~
  - o *FN-IN-05: The system should allow a user to or escape key to pause a game.*
  - o *FN-IN-06: The system should allow a user to select the desired gravity level.*
  - o *FN-IN-07: The system should allow a user to click, hold, and drag the mouse in order to create a shot vector.*
- **Output**
  - o *FN-OUT-01: The system shall run in in a maximized window.*
  - o *FN-OUT-02: The system shall display a menu screen on the completion of system loading.*
  - o *FN-OUT-03: The system shall have a settings menu that users can choose various settings from.*
  - o *FN-OUT-04: The system shall display current health for both players in the form of text.*
  - o *FN-OUT-05: The system shall display player sprites.*
  - o *FN-OUT-06: The system shall display the arrow as it moves.*
  - o *FN-OUT-07: The system shall be turned base, and only allow gameplay by the player whose current turn it is.*
  - ~~o *FN-OUT-08: The system shall display the correct texture under the player based on choice*~~
  - o *FN-OUT-09: The system shall display the correct color player texture based on choice*
  - o *FN-OUT-10: The system shall display the player's chosen name.*
  - o *FN-OUT-11: ~~The system shall display the model animations for shots, hits, and blood.~~*

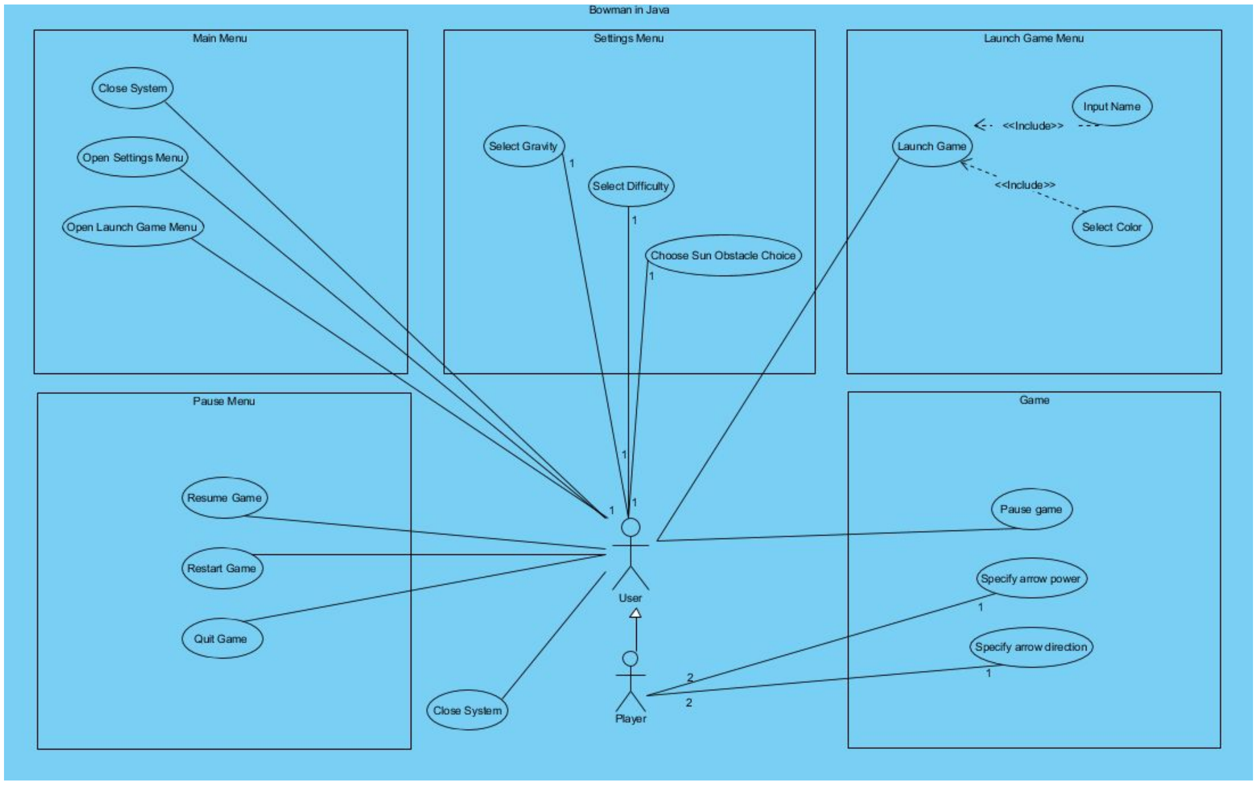
- *FN-OUT-12: The system shall display a map texture - will be decided upon at a later point in time.*
- ~~*FN-OUT-13: The system shall display necessary logos.*~~
- *FN-OUT-14: The system shall display the shot's vector information.*

- **Process**
  - *FN-PROC-01: The system shall run on a Windows, ~~Mac, or Linux/GNU~~ based operating system supporting Java SE 10.*
  - *FN-PROC-02: The system should allow use of the left mouse button and mouse movements from a USB mouse or laptop trackpad.*
  - *FN-PROC-03: The camera shall follow the the current shot or on the player when there isn't a shot*
  - *FN-PROC-04: The system should create a vector based on the angle and magnitude of the mouse input.*
  - *FN-PROC-05: The system should calculate all arrow movements based on kinematics.*
  - *FN-PROC-06: The system shall calculate the distance between players.*

Non-Functional

- **Input**
  - *NFR-IN-01: The system shall calculate an angle as well as velocity vector as a user clicks and drags during their turn pre-shot.*
  - *NFR-IN-02: The system shall listen for arrow impact to player model objects.*
- **Output**
  - *NFR-OUT-01: The system shall load to the menu screen in under 10 seconds.*
  - *NFR-OUT-02: The system shall distribute damage based on where the arrow hits on the height of the player model.*
  - *NFR-OUT-03: The system shall respond to player input within 0.5 seconds.*
  - *NFR-OUT-04: The system must respond to player damage within 0.5 seconds.*
- **Process**
  - *NFR-PROC-01: All Java programming conventions should be followed. See http://www.oracle.com/technetwork/java/javaee/downloads/codeconvtoc-136057.html for details.*
  - *NFR-PROC-02: The system should calculate a projectile motion path based upon a given angle and velocity vector.*
  - *NFR-PROC-03: The system shall calculate player damage based upon player player hit height as well as difficulty.*

- *NFR-PROC-09: The system shall keep track of the turns by using pointers for the turn player and other player.*
- *NFR-PROC-04: The system shall keep track of player health over the period of a game.*
- *NFR-PROC-05: The system shall utilize less than 50% single core performance at any given time.*
- ~~*NFR-PROC-06: The system shall stay above a framerate of at least 30fps at any point after the menu screen has loaded.*~~
- *NFR-PROC-07: The system shall run on a Windows, ~~Mac, or Linux/GNU~~ based operating system supporting Java SE 10.*
- *NFR-PROC-08: The system shall keep track of the number of arrows shot in a queue.*
- *NFR-PROC-10: The system shall only allow shooting for the player whose turn it is; the turn changes when the arrow hits the ground or the target.*
- *NFR-PROC-11: The system shall only allow a certain amount of arrows to remain for performance reasons.*
- *NFR-PROC-12: Calculate distance between players based on difficulty with random variance based on difficulty.*



Use Cases

Number	1
System	Bowman in Java (subsystem: main menu)
Name	Close System
Primary Actor(s)	User
Description	Close the system. Program exits.
Preconditions	
Post-conditions	The program exits.
Trigger	User clicks the quit button in the main menu
Basic Flow	1. Select Quit in main menu 2. System quits
Alternate Flow(s)	A.1. Select quit while in a game
Exception Flow(s)	

Extensions	

Number	2
System	Bowman in Java
Name	Close System
Primary Actor(s)	User
Description	The user presses the red X close button anywhere in the system. The system exits with no confirmation message.
Preconditions	The user is anywhere in the system and presses the Windows red X button.
Post-conditions	The system exits
Trigger	Press Windows close button
Basic Flow	1. Select Windows close button 2. Program exits
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

Number	3
System	Bowman in Java (subsystem: main menu)
Name	Open Settings menu
Primary Actor(s)	User
Description	User leaves the main menu and opens the settings menu
Preconditions	Main menu is opened.
Post-conditions	Settings menu is open
Trigger	User clicks the settings button in the main menu
Basic Flow	1. Be in main menu 2. Click settings button 3. Settings menu is opened

Alternate Flow(s)	
Exception Flow(s)	
Extensions	

Number	4
System	Bowman in Java (subsystem: settings menu)
Name	Select Gravity
Primary Actor(s)	User
Description	User selects game gravity. Has slider to choose from 0 to 10.
Preconditions	The user must currently have the settings menu open.
Post-conditions	Gravity level is set.
Trigger	The user changes the gravity slider.
Basic Flow	1. Settings menu is open. 2. Change the gravity slider 3. Gravity is changed.
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

Number	5
System	Bowman in Java (subsystem: settings menu)
Name	Select Difficulty
Primary Actor(s)	User
Description	User selects game difficulty. Can choose from easy, normal, or hard.
Preconditions	The user must currently have the settings menu open.
Post-conditions	Game difficulty is set
Trigger	The user clicks the easy, normal, or hard button.

Basic Flow	1. Settings menu is opened 2. User selects a difficulty 3. Difficulty is changed based on user input.
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

Number	6
System	Bowman in Java (subsystem: settings menu)
Name	Choose Sun Obstacle Choice
Primary Actor(s)	User
Description	User modifies a checkbox to determine if the sun entity will act as an obstacle in game.
Preconditions	The user must currently have the settings menu open.
Post-conditions	Boolean for sun as obstacle is set.
Trigger	The user modifies the checkbox.
Basic Flow	1. Settings menu is open 2. Player checks the sun obstacle check box 3. Sun is now an obstacle.
Alternate Flow(s)	A1.1 Settings menu is open A1.2 Player unchecks the sun obstacle check box A1.3 Sun is no longer an obstacle.
Exception Flow(s)	
Extensions	

Number	7
System	Bowman in Java (subsystem: launch game menu)
Name	Select Color

Primary Actor(s)	User
Description	User selects a color for their character model
Preconditions	User must be in launch game menu.
Post-conditions	User has a color.
Trigger	Select a color from the drop down.
Basic Flow	1. Player is in launch game menu 2. Player selects the color drop down 3. Player selects a color from drop down. 4. Player model has a color.
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player selects main menu button. E1.2 Player returns to main menu
Extensions	

Number	8
System	Bowman in Java (subsystem: launch game menu)
Name	Input name
Primary Actor(s)	User
Description	User inputs a string to be their name.
Preconditions	User must be in launch game menu.
Post-conditions	User has a name.
Trigger	Type in a name.
Basic Flow	1. Player is in launch game menu 2. Player types name into the provided text box. 3. Player model has a name
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player selects main menu button. E1.2 Player returns to main menu.
Extensions	

Number	9
System	Bowman in Java (subsystem: pause menu)
Name	Resume game
Primary Actor(s)	Player
Description	Player returns to game from the pause menu.
Preconditions	Game must be paused
Post-conditions	Game is resumed
Trigger	Select resume game button in pause menu.
Basic Flow	1. Player is in pause menu. 2. Player selects resume game 3. Game is resumed
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player selects quit game E1.2 Game is returned to main menu E2.1 Player selects restart game E2.2 Game is restarted.
Extensions	

Number	10
System	Bowman in Java (subsystem: pause menu)
Name	Quit game
Primary Actor(s)	Player
Description	Player quits the game from the pause menu.
Preconditions	Game must be paused
Post-conditions	Game is returned to the main menu
Trigger	Select quit game button in pause menu.
Basic Flow	1. Player is in pause menu 2. Player selects quit game

	3. Game is returned to main menu
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player selects resume game E1.2 Game is resumed E2.1 Player selects restart game E2.2 Game is restarted.
Extensions	

Number	11
System	Bowman in Java (subsystem: pause menu)
Name	Restart game
Primary Actor(s)	Player
Description	The game goes back to the beginning, starting with player one's turn and setting both players back to max health.
Preconditions	Game must be paused
Post-conditions	Game is restarted.
Trigger	Select restart game button in pause menu.
Basic Flow	1. Player is in pause menu 2. Player selects restart game 3. Game is restarted.
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player selects resume game E1.2 Game is resumed E2.1 Player selects quit game E2.2 Game is returned to main menu
Extensions	

Number	12

System	Bowman in Java (subsystem: game)
Name	Pause game
Primary Actor(s)	Player
Description	Player pauses the game
Preconditions	Player is currently in a game.
Post-conditions	The game is paused, and the menu is activated.
Trigger	Press the escape key on the keyboard.
Basic Flow	1. Player is in game 2. Player presses escape key 3. Pause menu is opened.
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

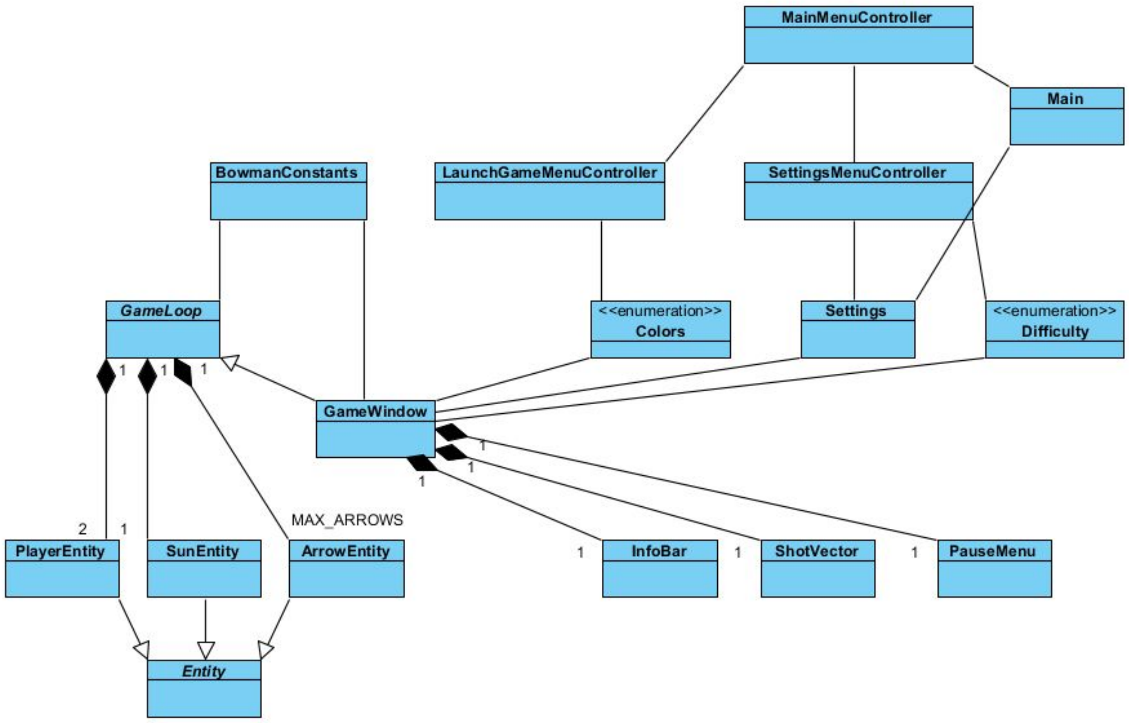
Number	13
System	Bowman in Java (subsystem: game)
Name	Specify arrow power
Primary Actor(s)	Player
Description	A magnitude is input to the system in order to help calculate the shot vector.
Preconditions	Player is currently in a game, and it is player's turn.
Post-conditions	Player fires an arrow
Trigger	Player clicks, holds, and drags the mouse a certain distance. System calculates the power based on mouse movement. This happens simultaneously with specifying arrow direction.
Basic Flow	1. Player is in game 2. Player clicks, holds, and drags mouse. 3. Power is calculated
Alternate Flow(s)	

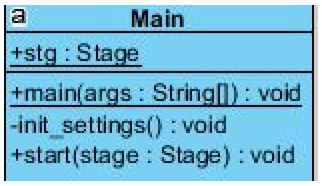
Exception Flow(s)	
Extensions	

Number	14
System	Bowman in Java (subsystem: game)
Name	Specify arrow direction
Primary Actor(s)	Player
Description	A direction is input to the system in order to help calculate the shot vector.
Preconditions	Player is currently in a game, and it is player's turn
Post-conditions	Player fires an arrow.
Trigger	Player clicks, holds, and drags the mouse in a direction. System calculates the direction based on mouse movement. This happens simultaneously with specifying arrow power.
Basic Flow	1. Player is in game 2. Player clicks, holds, and drags mouse. 3. Direction is calculated
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

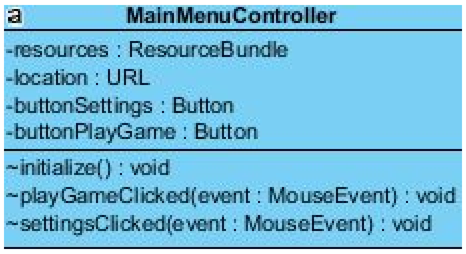
Number	15
System	Bowman in Java (subsystem: main menu)
Name	Open launch game menu
Primary Actor(s)	User
Description	Opens the launch game menu.
Preconditions	User is in main menu
Post-conditions	Launch game menu is opened
Trigger	Select the start game button in main menu.
Basic Flow	1. Be in main menu 2. Select start game button 3. Launch game menu is opened.
Alternate Flow(s)	
Exception Flow(s)	
Extensions	

Number	16
System	Bowman in Java (subsystem: launch game menu)
Name	Launch game
Primary Actor(s)	User
Description	User launches the game
Preconditions	Player is in launch game menu, and color and name is filled in.
Post-conditions	Game is started.
Trigger	User selects start game button
Basic Flow	1. User is in launch game menu 2. User selects start game button 3. Game is started
Alternate Flow(s)	
Exception Flow(s)	E1.1 Player's color is not selected E1.2 Alert is shown saying color must be selected. E2.1 Name text box is empty. E2.2 Alert is shown saying name must be provided.
Extensions	

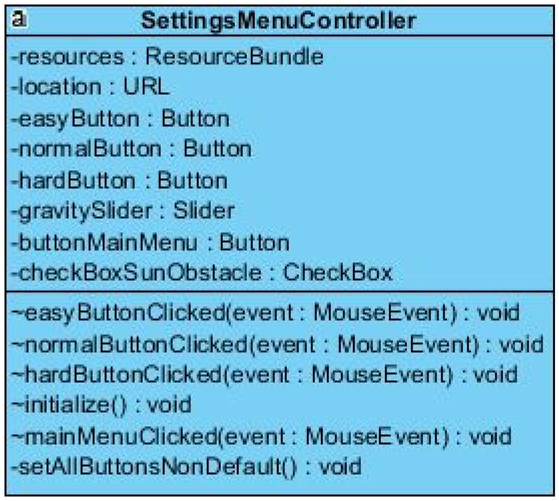

MainMenuController
Main
BowmanConstants
LaunchGameMenuController
SettingsMenuController
GameLoop
<<enumeration>> Colors
Settings
<<enumeration>> Difficulty
GameWindow
1
1
1
1
1
1
2
1
MAX_ARROWS
PlayerEntity
SunEntity
ArrowEntity
1
InfoBar
1
ShotVector
1
PauseMenu
Entity



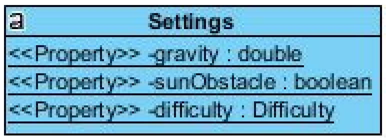
Description:
Handles launching the program.


MainMenuController
-resources : ResourceBundle
-location : URL
-buttonSettings : Button
-buttonPlayGame : Button
~initialize() : void
~playGameClicked(event : MouseEvent) : void
~settingsClicked(event : MouseEvent) : void

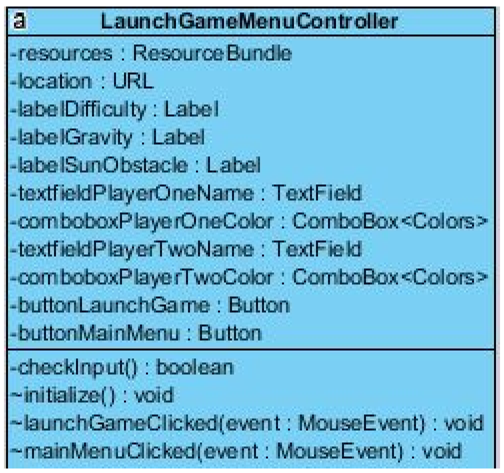
Description:
Handles when the user starts a game along with if they open the settings menu.


SettingsMenuController
-resources : ResourceBundle
-location : URL
-easyButton : Button
-normalButton : Button
-hardButton : Button
-gravitySlider : Slider
-buttonMainMenu : Button
-checkBoxSunObstacle : CheckBox
~easyButtonClicked(event : MouseEvent) : void
~normalButtonClicked(event : MouseEvent) : void
~hardButtonClicked(event : MouseEvent) : void
~initialize() : void
~mainMenuClicked(event : MouseEvent) : void
-setAllButtonsNonDefault() : void

Description:
Handles user changing settings to their preference. Difficulty along with gravity power and the Sun Obstacle checkbox are located here.


<<Property>> -gravity : double
<<Property>> -sunObstacle : boolean
<<Property>> -difficulty : Difficulty

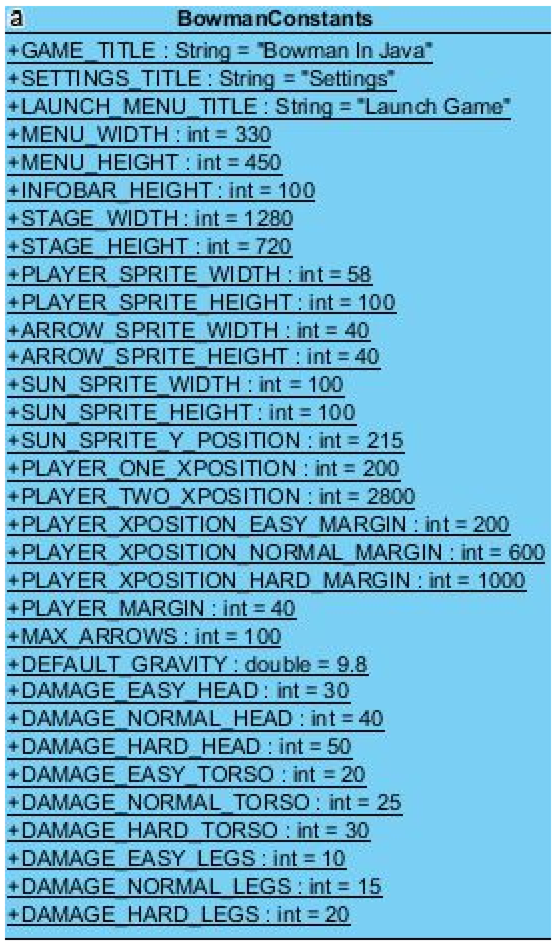
Description:
These properties are changed when the user changes them in settings.



```
LaunchGameMenuController
-resources : ResourceBundle
-location : URL
-labelDifficulty : Label
-labelGravity : Label
-labelSunObstacle : Label
-textfieldPlayerOneName : TextField
-comboboxPlayerOneColor : ComboBox<Colors>
-textfieldPlayerTwoName : TextField
-comboboxPlayerTwoColor : ComboBox<Colors>
-buttonLaunchGame : Button
-buttonMainMenu : Button
-checkInput() : boolean
~initialize() : void
~launchGameClicked(event : MouseEvent) : void
~mainMenuClicked(event : MouseEvent) : void
```

Description:

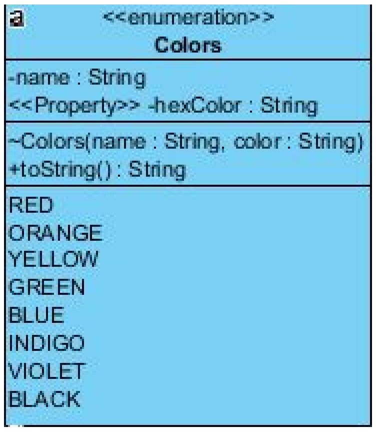
Handles events when the user is starting a game. Forces users to input names and colors.



```
BowmanConstants
+GAME_TITLE : String = "Bowman In Java"
+SETTINGS_TITLE : String = "Settings"
+LAUNCH_MENU_TITLE : String = "Launch Game"
+MENU_WIDTH : int = 330
+MENU_HEIGHT : int = 450
+INFOBAR_HEIGHT : int = 100
+STAGE_WIDTH : int = 1280
+STAGE_HEIGHT : int = 720
+PLAYER_SPRITE_WIDTH : int = 58
+PLAYER_SPRITE_HEIGHT : int = 100
+ARROW_SPRITE_WIDTH : int = 40
+ARROW_SPRITE_HEIGHT : int = 40
+SUN_SPRITE_WIDTH : int = 100
+SUN_SPRITE_HEIGHT : int = 100
+SUN_SPRITE_Y_POSITION : int = 215
+PLAYER_ONE_XPOSITION : int = 200
+PLAYER_TWO_XPOSITION : int = 2800
+PLAYER_XPOSITION_EASY_MARGIN : int = 200
+PLAYER_XPOSITION_NORMAL_MARGIN : int = 600
+PLAYER_XPOSITION_HARD_MARGIN : int = 1000
+PLAYER_MARGIN : int = 40
+MAX_ARROWS : int = 100
+DEFAULT_GRAVITY : double = 9.8
+DAMAGE_EASY_HEAD : int = 30
+DAMAGE_NORMAL_HEAD : int = 40
+DAMAGE_HARD_HEAD : int = 50
+DAMAGE_EASY_TORSO : int = 20
+DAMAGE_NORMAL_TORSO : int = 25
+DAMAGE_HARD_TORSO : int = 30
+DAMAGE_EASY_LEGS : int = 10
+DAMAGE_NORMAL_LEGS : int = 15
+DAMAGE_HARD_LEGS : int = 20
```

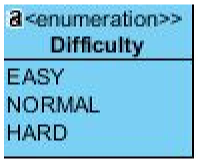
Description:

Constants used throughout the program.



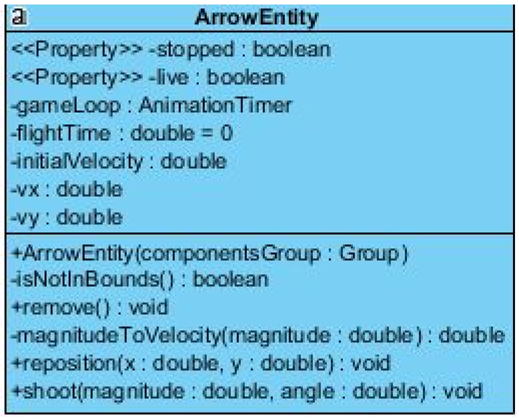
Description:

Enum for Colors.



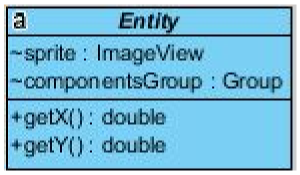
Description:

Enum for difficulty.


ArrowEntity
<<Property>> -stopped : boolean
<<Property>> -live : boolean
-gameLoop : AnimationTimer
-flightTime : double = 0
-initialVelocity : double
-vx : double
-vy : double
+ArrowEntity(componentsGroup : Group)
-isNotInBounds() : boolean
+remove() : void
-magnitudeToVelocity(magnitude : double) : double
+reposition(x : double, y : double) : void
+shoot(magnitude : double, angle : double) : void

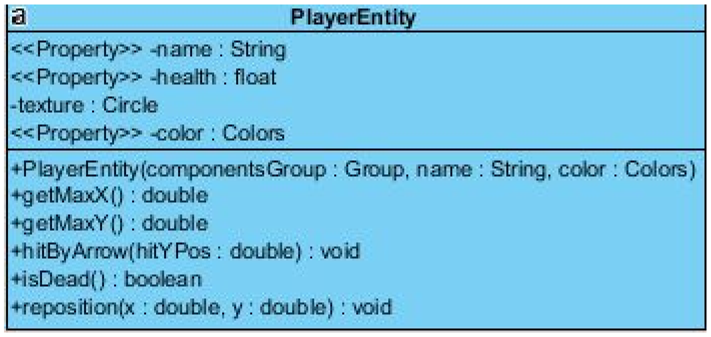
Description:

Handles the arrow entity. Also handles moving the arrow while the game loop is running.

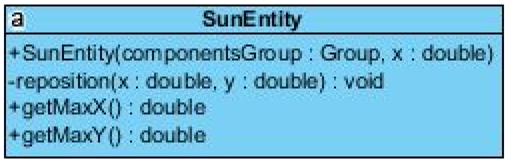


Description:

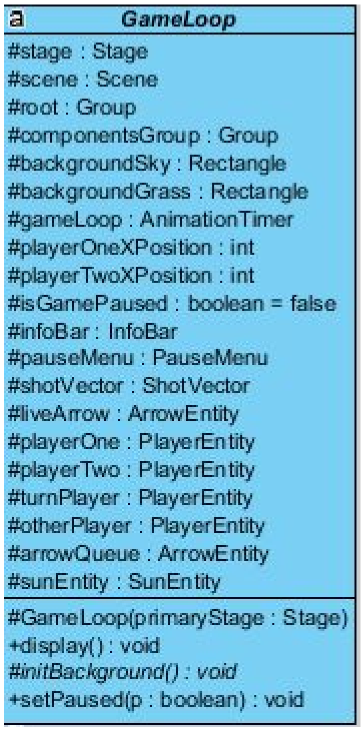
Superclass entity for all entities.


PlayerEntity
<<Property>> -name : String
<<Property>> -health : float
-texture : Circle
<<Property>> -color : Colors
+PlayerEntity(componentsGroup : Group, name : String, color : Colors)
+getMaxX() : double
+getMaxY() : double
+hitByArrow(hitYPos : double) : void
+isDead() : boolean
+reposition(x : double, y : double) : void

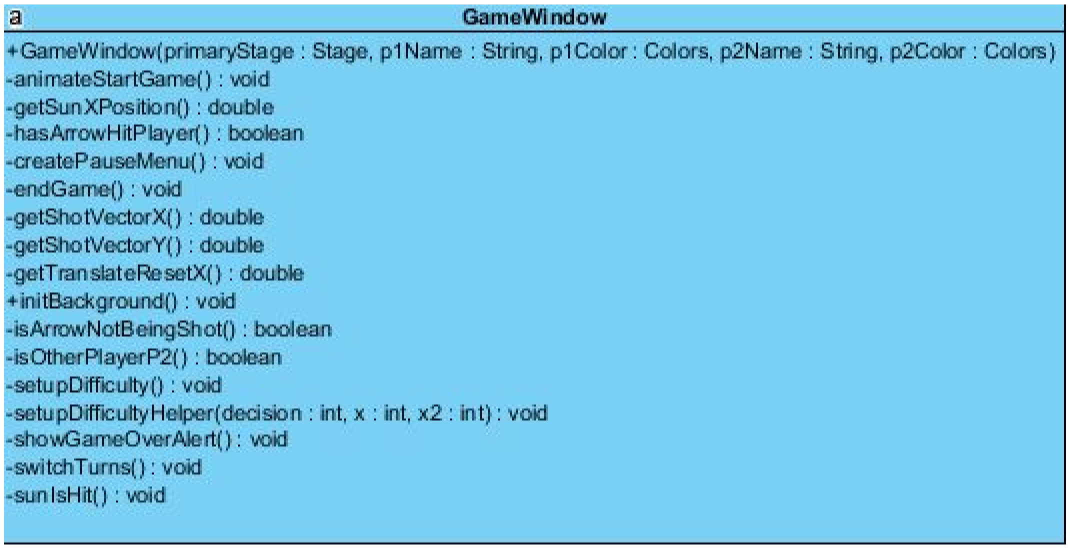
Description:
Handles player entities. Handles player related events and when they are hit by enemy arrows.


SunEntity
+SunEntity(componentsGroup : Group, x : double)
-reposition(x : double, y : double) : void
+getMaxX() : double
+getMaxY() : double

Description:
Handles creating the sun entity.


GameLoop
#stage : Stage
#scene : Scene
#root : Group
#componentsGroup : Group
#backgroundSky : Rectangle
#backgroundGrass : Rectangle
#gameLoop : AnimationTimer
#playerOneXPosition : int
#playerTwoXPosition : int
#isGamePaused : boolean = false
#infoBar : InfoBar
#pauseMenu : PauseMenu
#shotVector : ShotVector
#liveArrow : ArrowEntity
#playerOne : PlayerEntity
#playerTwo : PlayerEntity
#turnPlayer : PlayerEntity
#otherPlayer : PlayerEntity
#arrowQueue : ArrowEntity
#sunEntity : SunEntity
#GameLoop(primaryStage : Stage)
+display() : void
#initBackground() : void
+setPaused(p : boolean) : void

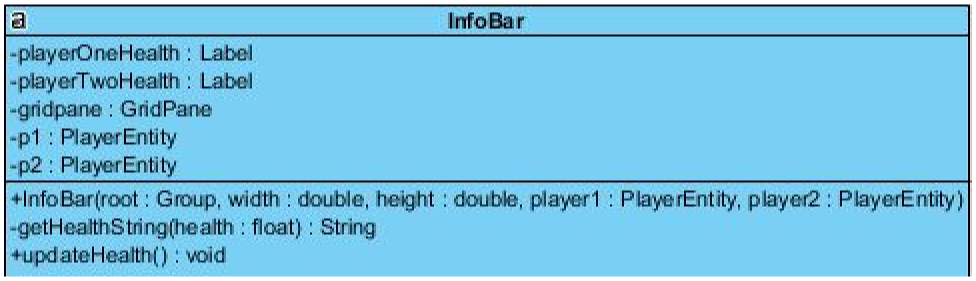
Description:
The class that handles running the game. Runs until a player pauses the game or a player quits or a player dies.



```
GameWindow
+GameWindow(primaryStage : Stage, p1Name : String, p1Color : Colors, p2Name : String, p2Color : Colors)
-animateStartGame() : void
-getSunXPosition() : double
-hasArrowHitPlayer() : boolean
-createPauseMenu() : void
-endGame() : void
-getShotVectorX() : double
-getShotVectorY() : double
-getTranslateResetX() : double
+initBackground() : void
-isArrowNotBeingShot() : boolean
-isOtherPlayerP2() : boolean
-setupDifficulty() : void
-setupDifficultyHelper(decision : int, x : int, x2 : int) : void
-showGameOverAlert() : void
-switchTurns() : void
-sunIsHit() : void
```

Description:

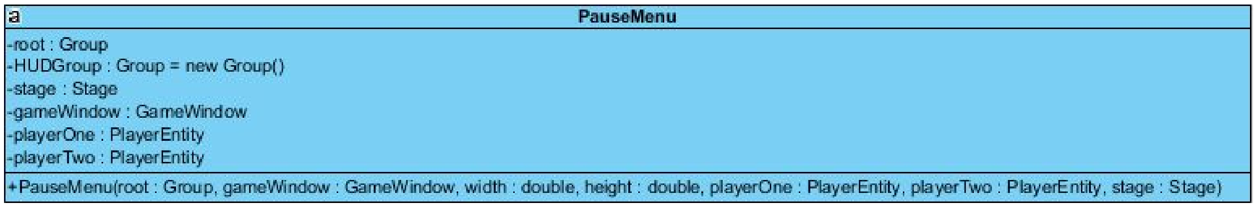
Class handles game window related events like pause and alerts. Handles when the arrows hit the Sun. Handles startup animations.



```
InfoBar
-playerOneHealth : Label
-playerTwoHealth : Label
-gridpane : GridPane
-p1 : PlayerEntity
-p2 : PlayerEntity
+InfoBar(root : Group, width : double, height : double, player1 : PlayerEntity, player2 : PlayerEntity)
-getHealthString(health : float) : String
+updateHealth() : void
```

Description:

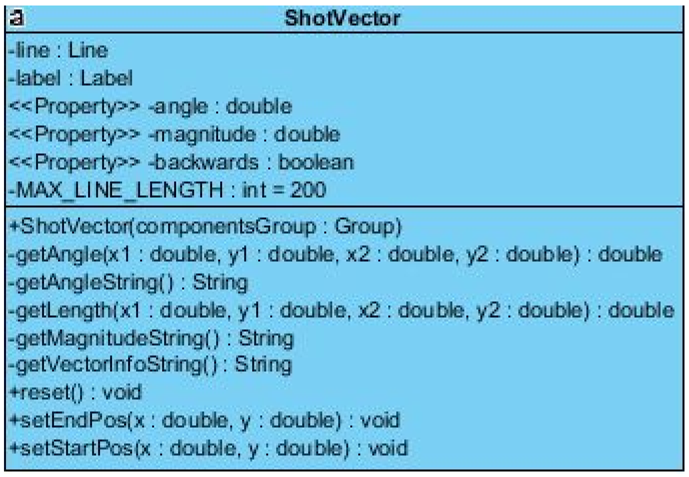
Class that handles the infobar that is created at the top of the game window.



```
PauseMenu
-root : Group
-HUDGroup : Group = new Group()
-stage : Stage
-gameWindow : GameWindow
-playerOne : PlayerEntity
-playerTwo : PlayerEntity
+PauseMenu(root : Group, gameWindow : GameWindow, width : double, height : double, playerOne : PlayerEntity, playerTwo : PlayerEntity, stage : Stage)
```

Description:

Class that handles when either player pauses the game.

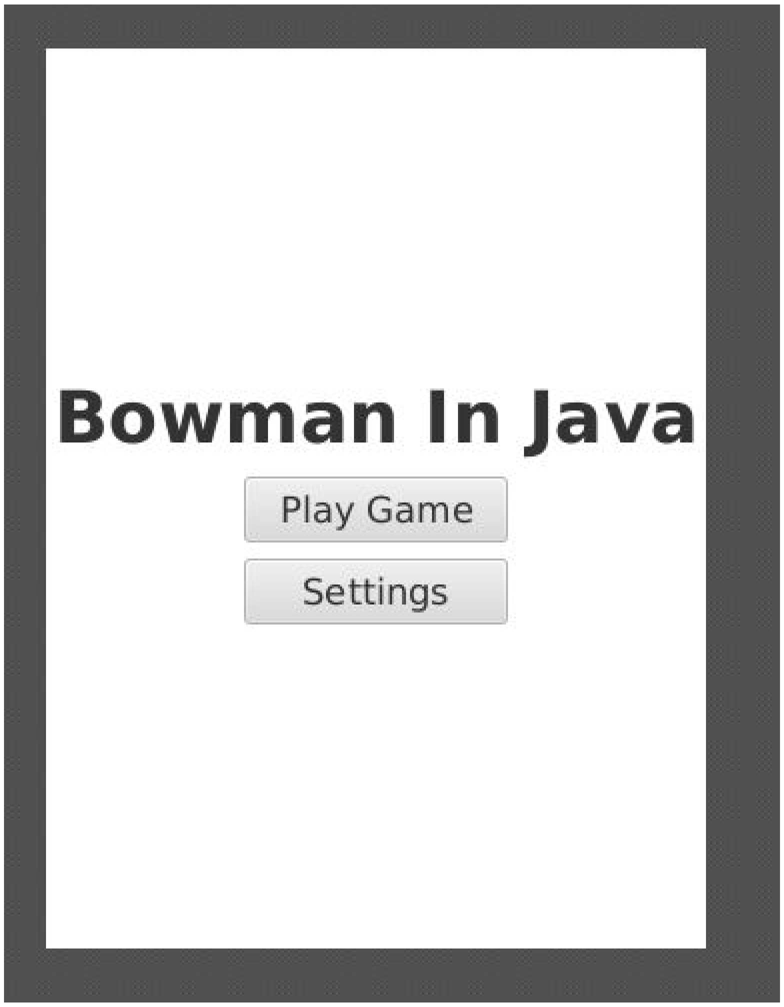

ShotVector
-line : Line
-label : Label
<<Property>> -angle : double
<<Property>> -magnitude : double
<<Property>> -backwards : boolean
-MAX_LINE_LENGTH : int = 200
+ShotVector(componentsGroup : Group)
-getAngle(x1 : double, y1 : double, x2 : double, y2 : double) : double
-getAngleString() : String
-getLength(x1 : double, y1 : double, x2 : double, y2 : double) : double
-getMagnitudeString() : String
-getVectorInfoString() : String
+reset() : void
+setEndPos(x : double, y : double) : void
+setStartPos(x : double, y : double) : void

Description:

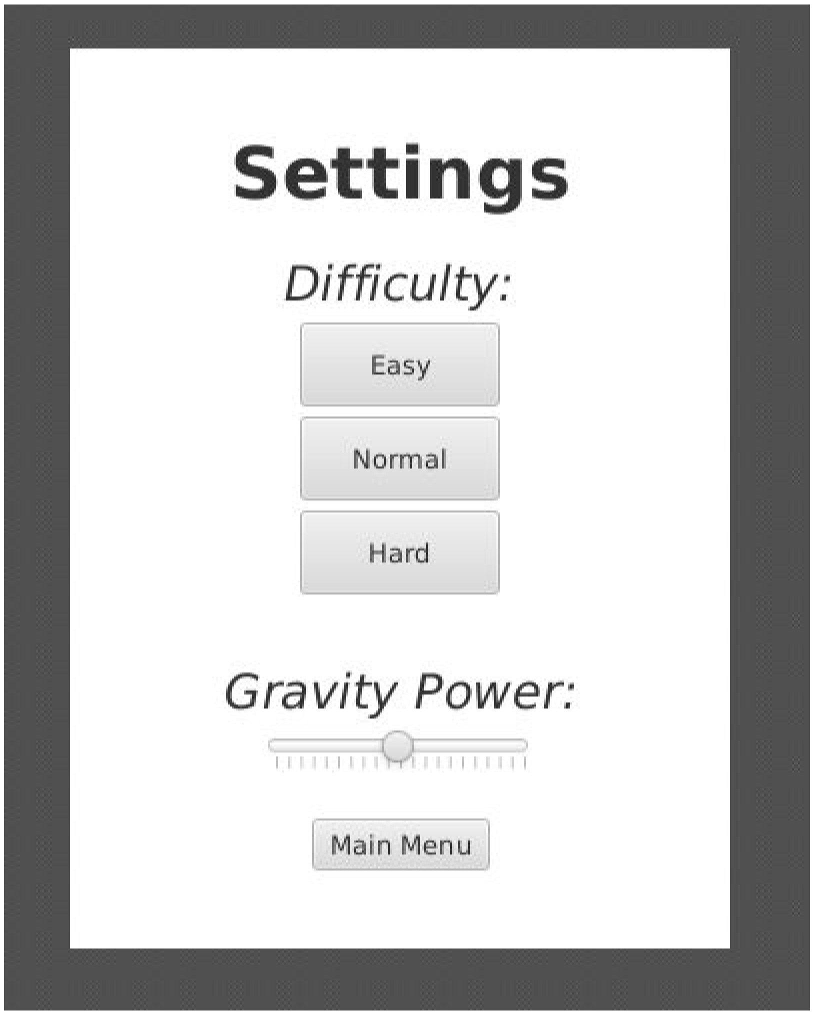
Class that handles displaying shot vector when the player goes to fire an arrow.

UI Mockups

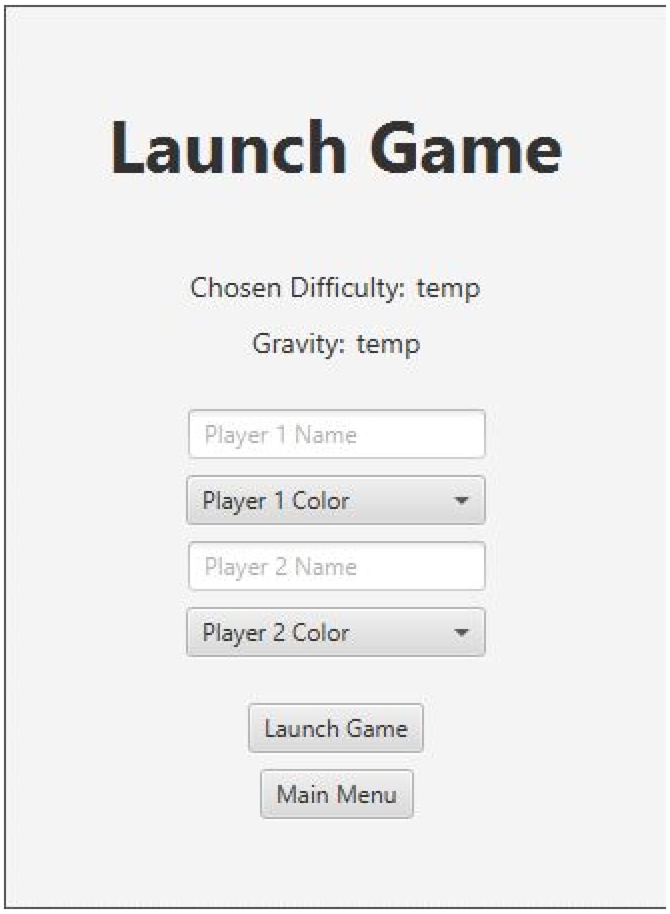
Main Menu



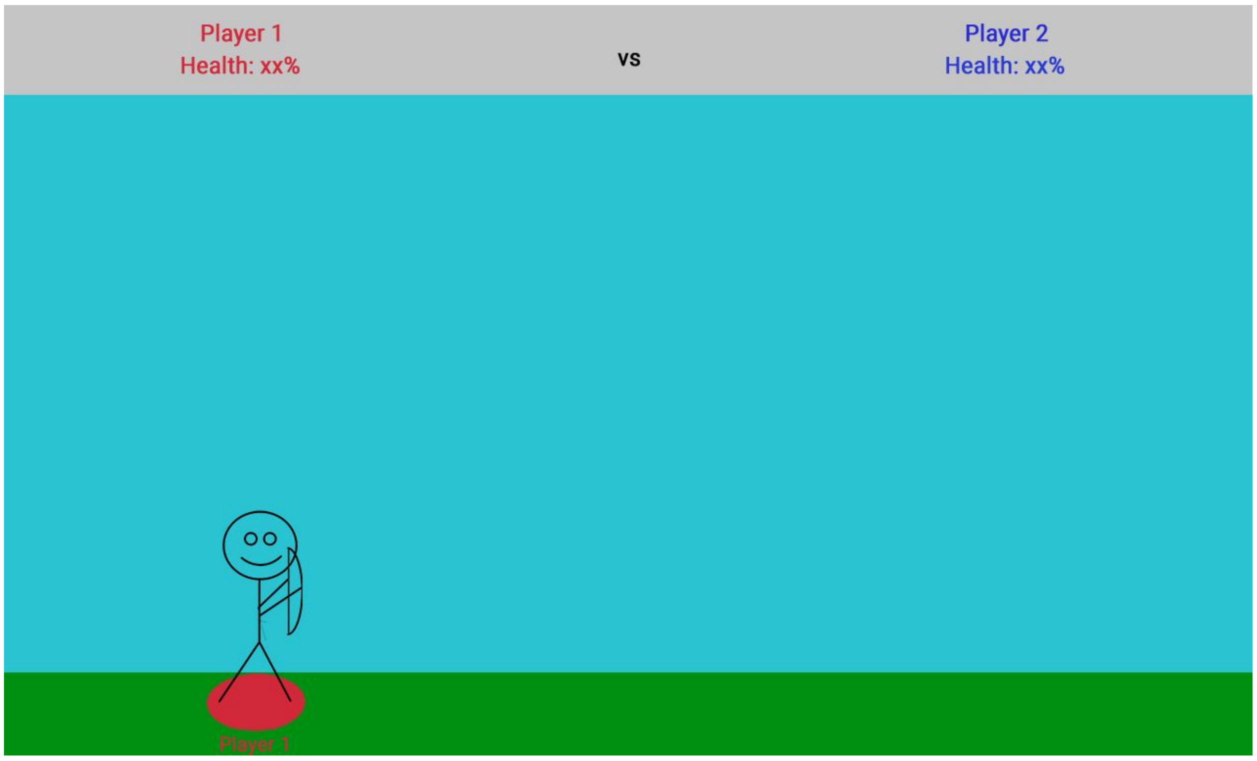
Settings Menu



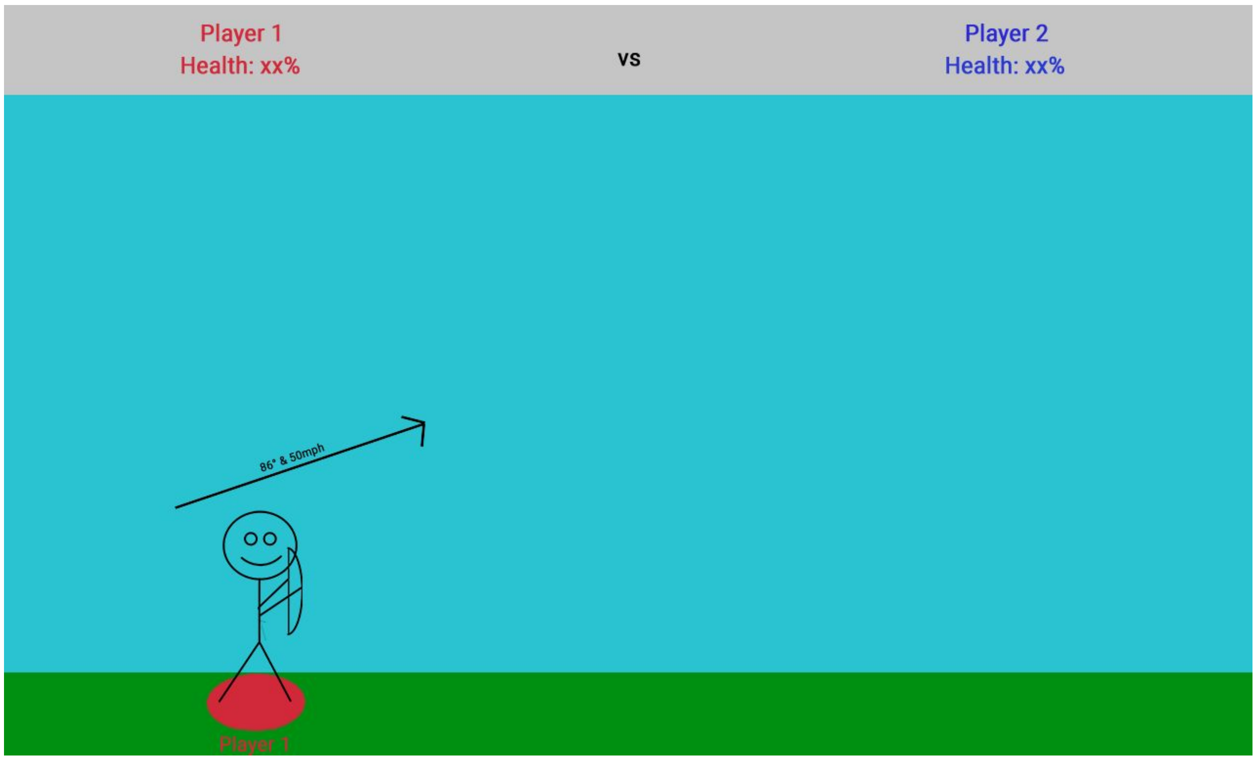
Launch Game Menu



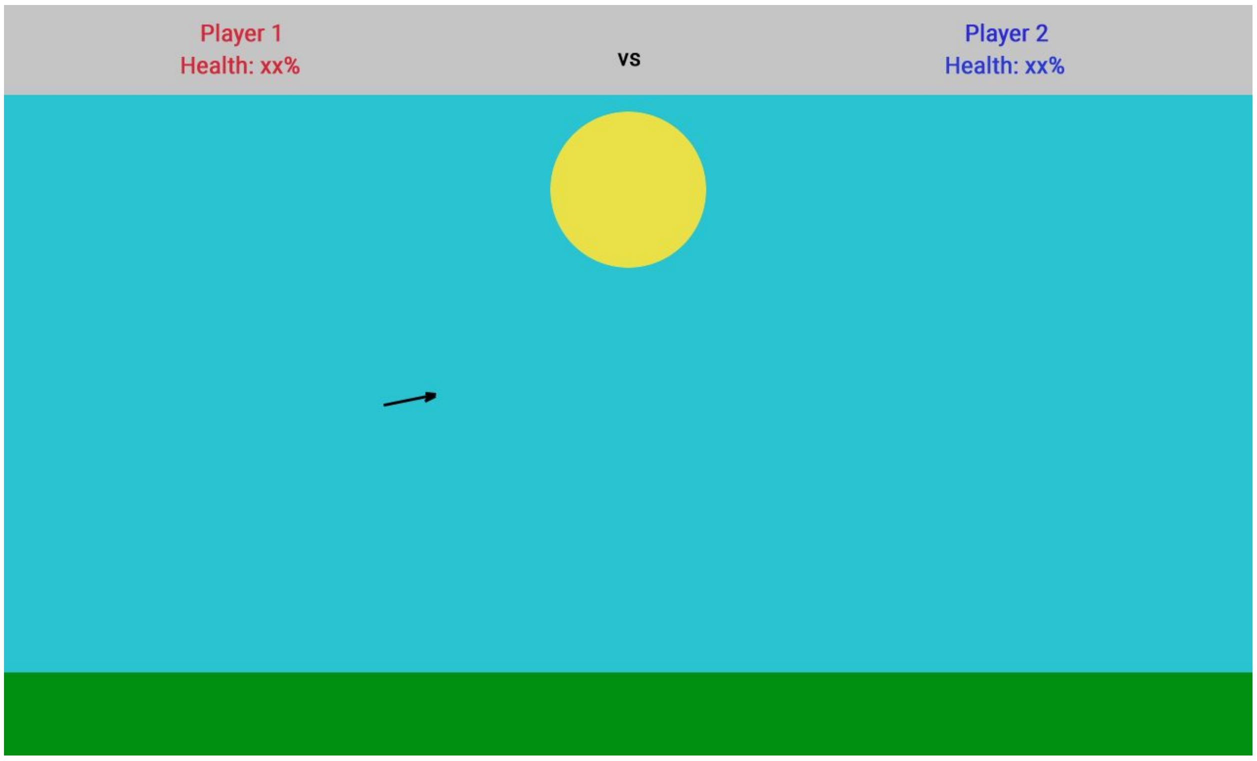
Game Window - Player 1



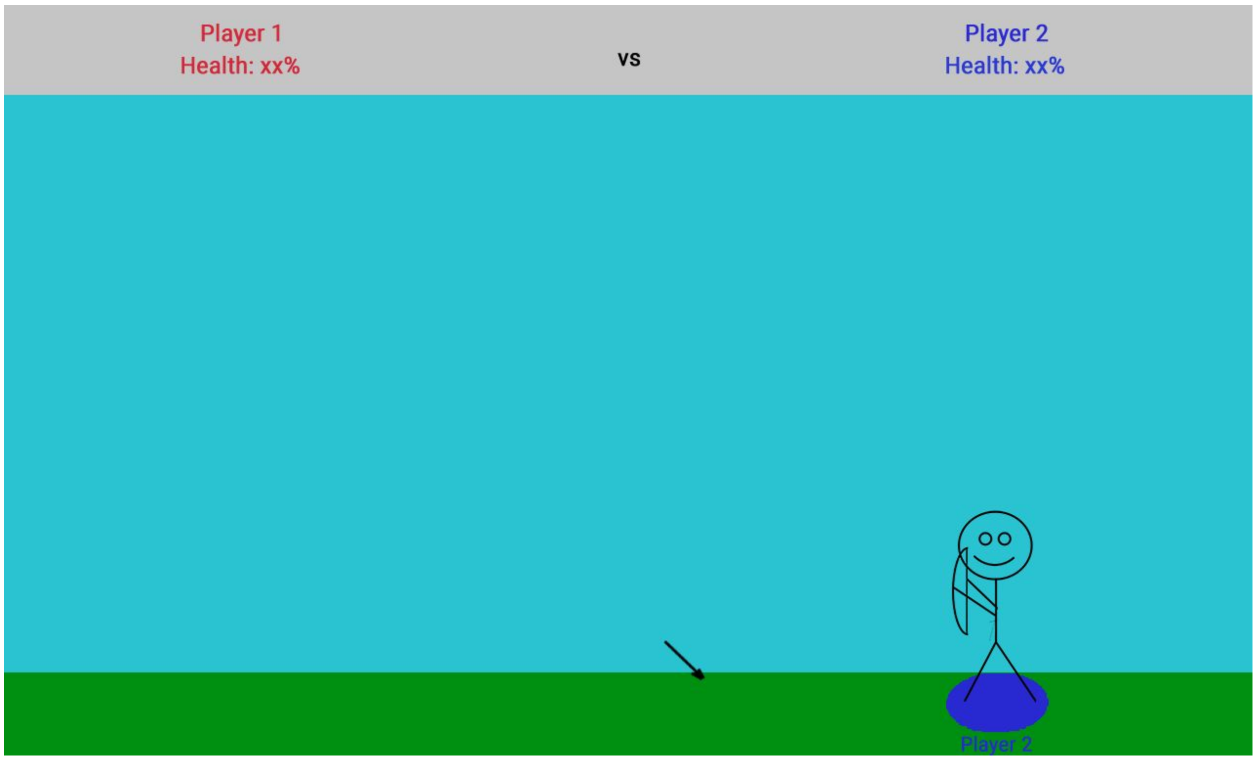
Game Window - Player 1 Shooting



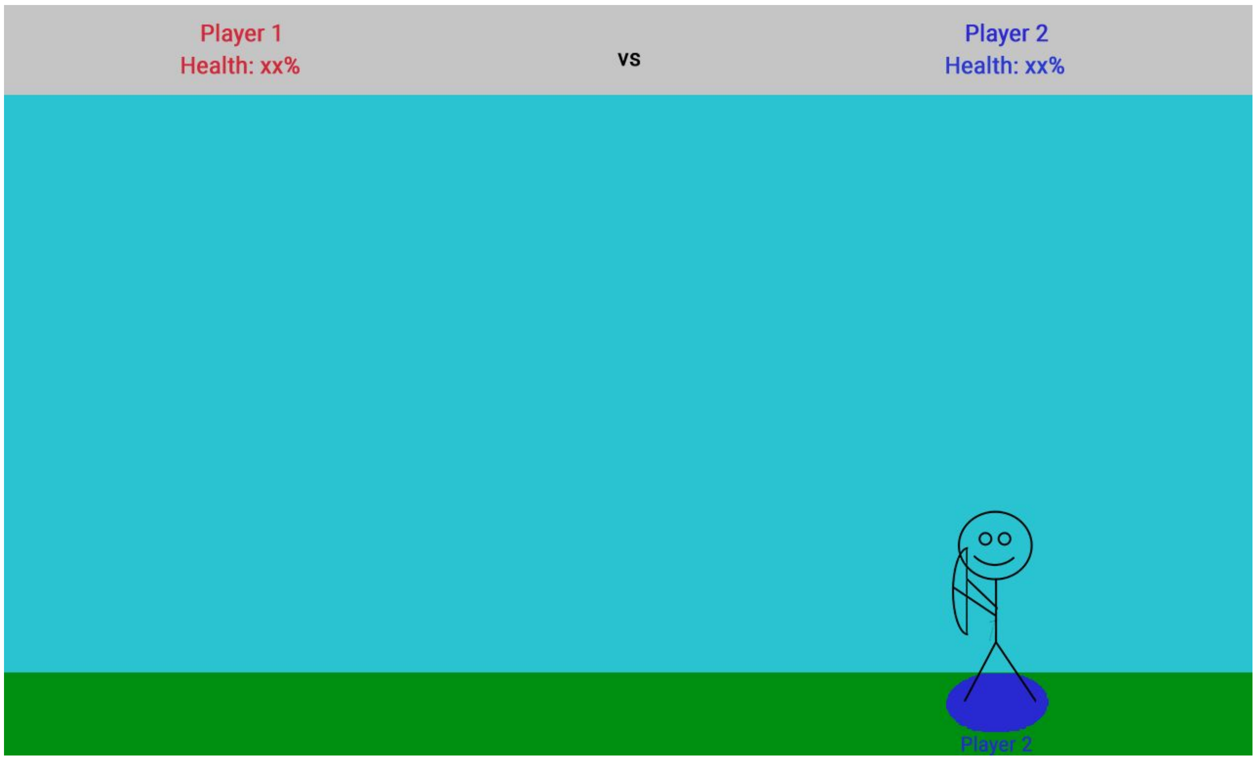
Game Window - Arrow in flight



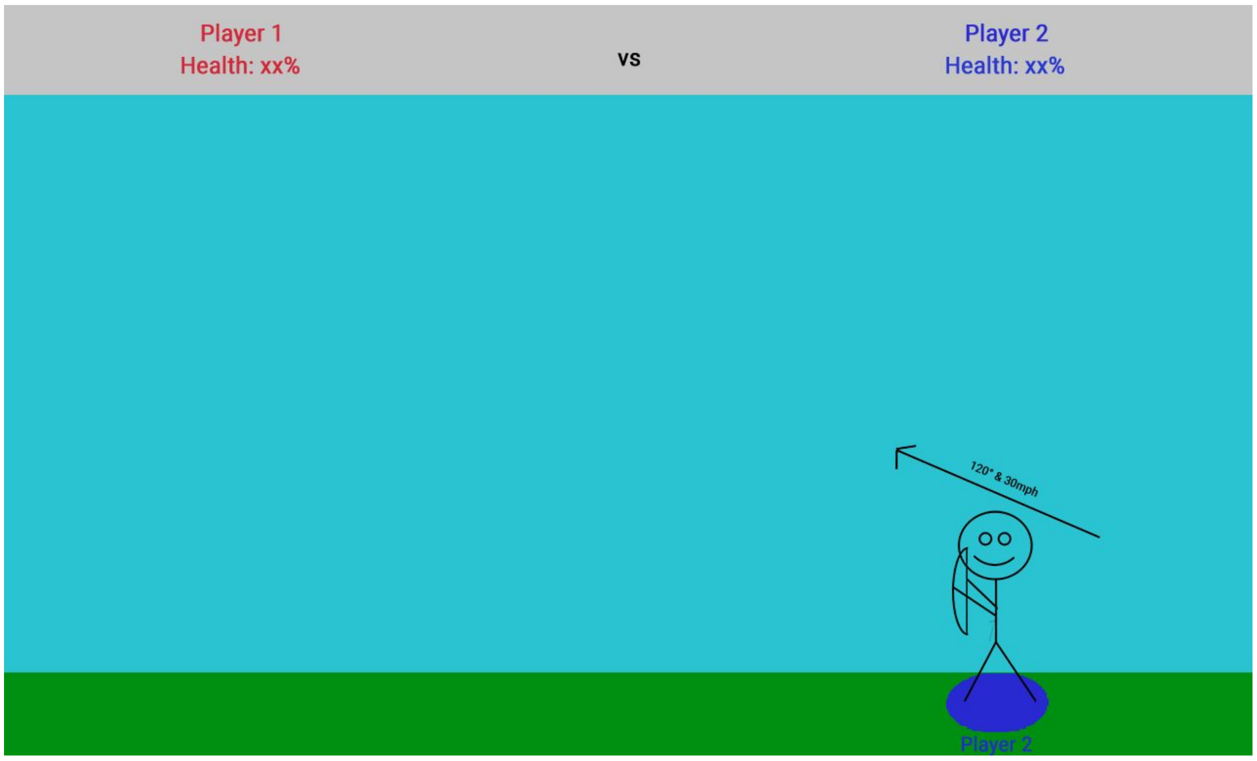
Game Window - Arrow in ground



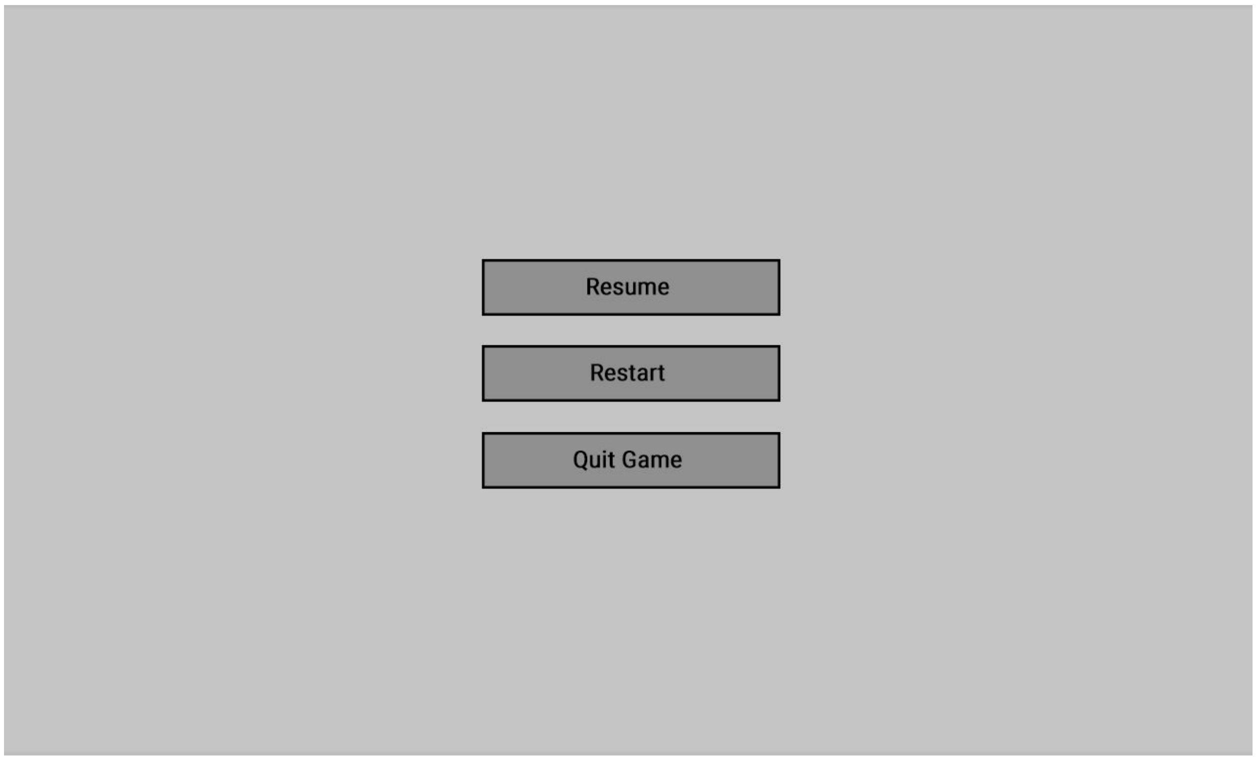
Game Window - Player 2



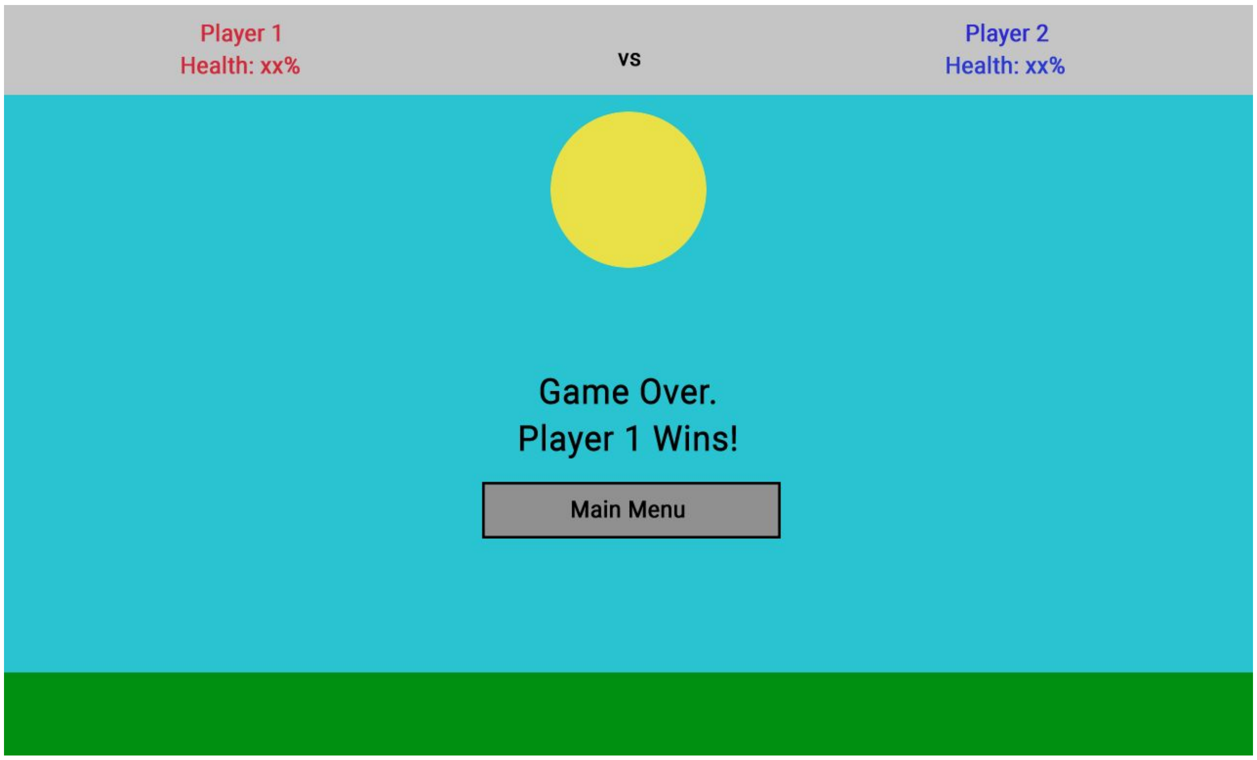
Game Window - Player 2 Shooting



Game Window - Pause Menu



Game Window - Game Over



Test Designs

1 Title

Design Document for Bowman in Java

2 Team

Austin FitzGerald, Tim Collier, Noah Beilke

3 Test Design

Test Case ID	*T001*
Purpose	Selecting the difficulty
Pre-conditions	Be in the settings menu
Inputs	Select button for prefered difficulty
Expected Outputs	The button changes colors so the user knows it is clicked.
Post-conditions	Difficulty is adjusted accordingly
Design Technique	Code Inspection, Visual Inspection

Test Case ID	*T002*
Purpose	Assign each player a name
Pre-conditions	Be in the game launching menu
Inputs	Each player types in their name, the textbox accepts all characters.
Expected Outputs	Textbox will update
Post-conditions	Each player is assigned a name
Design Technique	Code Inspection

Test Case ID	*T003*
Purpose	Assign each player a color and texture
Pre-conditions	Be in the game launching menu
Inputs	Each player selects a color from the dropdown
Expected Outputs	Dropdown will update to show selected color
Post-conditions	The player will be assigned a color, their texture will be that color
Design Technique	Code Inspection

Test Case ID	*T004*
Purpose	Open the pause-menu
Pre-conditions	The user is currently in a game
Inputs	Click the escape key
Expected Outputs	The pause-menu is displayed
Post-conditions	The game is paused and no player-specific inputs are allowed
Design Technique	Visual inspection

Test Case ID	*T005*
Purpose	Un-pause the game from the pause-menu
Pre-conditions	The user is currently in the pause-menu
Inputs	Click the “Resume” button
Expected Outputs	The pause-menu closes
Post-conditions	The game is resumed, player inputs are restored
Design Technique	Visual inspection

~~Test Case ID~~	~~*T006*~~
~~Purpose~~	~~Display animations~~
~~Pre-conditions~~	~~Arrow hits a player~~
~~Inputs~~	~~Arrow is shot hitting the other player~~
~~Expected Outputs~~	~~Blood animations are displayed~~
~~Post-conditions~~	~~The other player’s turn~~
~~Design Technique~~	~~Visual inspection~~

Test Case ID	*T007*
Purpose	Displaying vector information
Pre-conditions	Be in the game, be a player’s turn
Inputs	Draw back the arrow with the mouse
Expected Outputs	As the angle and magnitude of click length and direction changes, display angle and magnitude accordingly
Post-conditions	The angle and magnitude are displayed near the player model
Design Technique	Visual inspection, code inspection

Test Case ID	*T008*
Purpose	Displaying arrow movement
Pre-conditions	The arrow getting pulled back
Inputs	Release the arrow
Expected Outputs	Arrow is released, flies with the measured angle and magnitude
Post-conditions	The arrow is displayed flying along the calculated path
Design Technique	Visual inspection

Test Case ID	*T009*
Purpose	Displaying main-menu
Pre-conditions	The system must be loaded
Inputs	none
Expected Outputs	Main menu is opened
Post-conditions	Users can now select main menu options
Design Technique	Visual inspection

Test Case ID	*T010*
Purpose	Displaying initial game components
Pre-conditions	Be in the game launching menu
Inputs	Select the start game button
Expected Outputs	All game components should be displayed. This includes the game background, the information bar, and correct camera placement.
Post-conditions	The game begins.
Design Technique	Visual Inspection

Test Case ID	*T011*
Purpose	Display player models appropriate distance apart
Pre-conditions	Start game button was selected. Distance is calculated based on difficulty level
Inputs	none
Expected Outputs	Player models displayed calculated distance apart
Post-conditions	Game begins

Design Technique	Code Inspection, Visual Inspection

Test Case ID	*T012*
Purpose	System listening for arrow impacts
Pre-conditions	The game is started
Inputs	Arrow is fired
Expected Outputs	Returns true when the arrow hits another object
Post-conditions	Boolean goes back to false when turn changes to other player
Design Technique	Code Inspection

Test Case ID	*T013*
Purpose	System utilizes less than 50% of one core
Pre-conditions	System is running
Inputs	
Expected Outputs	
Post-conditions	CPU usage is under 50%
Design Technique	Visual Inspection

Test Case ID	*T014*
Purpose	Calculate arrow position
Pre-conditions	Arrow is drawn back
Inputs	Arrow is fired
Expected Outputs	The arrow follows the path calculated by a created function
Post-conditions	Arrow hits something
Design Technique	Code Inspection

Test Case ID	*T015*
Purpose	Calculate damage
Pre-conditions	Arrow hits player model
Inputs	None
Expected Outputs	Damage is calculated based on what part of the player model was hit, and then returned.
Post-conditions	New health is displayed

Design Technique	Code Inspection

Test Case ID	*T016*
Purpose	End game on player death
Pre-conditions	Arrow hits player model and player's health is decreased to zero or below
Inputs	none
Expected Outputs	Window is displayed: Game ends, other player wins
Post-conditions	Game ends, returns to main menu
Design Technique	Visual Inspection,

Test Case ID	*T017*
Purpose	Return to main menu on "Quit Game" chosen
Pre-conditions	Be in the game
Inputs	Click the quit game button
Expected Outputs	Returns to the main menu
Post-conditions	Be in the main menu.
Design Technique	Visual Inspection

Test Case ID	*T018*
Purpose	Restart the game on "Restart" chosen
Pre-conditions	Be in the game
Inputs	Click the restart button
Expected Outputs	Health is reset, and all arrow objects are cleared from the scene
Post-conditions	Game is restarted to the first player's turn
Design Technique	Visual Inspection

Test Case ID	*T019*
Purpose	Delete arrows in queue fashion when maximum arrow-count is reached
Pre-conditions	Maximum number of arrows are fired
Inputs	None
Expected Outputs	Queue deletes an arrow for each additional arrow fired and stored in the queue.
Post-conditions	The deleted arrow disappears from the game scene and memory
Design Technique	Visual Inspection, code inspection

Test Case ID	*T020*
Purpose	Change turns after a player shoots
Pre-conditions	Be in the game
Inputs	Player whose turn it is fires an arrow and arrow
Expected Outputs	When the arrow hits player mode, the turn changes.
Post-conditions	It is the other player’s turn.
Design Technique	Visual Inspection, code inspection

4 Traceability

Modified Traceability Table

Test Case	Requirement(s)
T001	FN-IN-01, FN-OUT-03,
T002	FN-IN-02, FN-IN-03
T003	FN-IN-04,
T004	FN-IN-05, NFR-OUT-03,
T005	FN-OUT-01,
~~T006~~	~~FN-OUT-11,~~
T007	FN-OUT-14, FN-PROC-04, FN-PROC-05, NFR-IN-01, NFR-PROC-02
T008	FN-OUT-06, FN-PROC-03, FN-PROC-05
T009	FN-OUT-02, FN-PROC-01, FN-PROC-02, NFR-PROC-07, ~~FN-OUT-13~~, NFR-PROC-06, NFR-OUT-01
T010	FN-OUT-12, FN-OUT-04, FN-OUT-05, FN-PROC-06
T011	FN-PROC-06, ~~FN-OUT-08~~, FN-OUT-09, FN-OUT-10, NFR-PROC-12
T012	NFR-IN-02, NFR-OUT-04,
T013	NFR-PROC-05
T014	FN-OUT-06, NFR-PROC-02, NFR-IN-01
T015	NFR-PROC-12, NFR-PROC-03
T016	NFR-PROC-04
T017	FN-IN-05, FN-PROC-02
T018	FN-PROC-02
T019	NFR-PROC-08, NFR-PROC-11
T020	FN-OUT-07, NFR-PROC-10, NFR-PROC-09

6 Glossary

Term	Definition
System	The program "Bowman in Java" which includes all its menus and windows.
Game	The system window of which the match is played that both *P1 and P2* interact with.
User	Used for whoever is currently using the mouse/keyboard. This term is used when it is not important who is doing the input. A user is not player-specific.
Location	A predetermined area on the map that represents the location of one of the players
Difficulty	The difficulty in the game. This changes how much damage is applied
Player	Users when they are playing the game
P1	Refers to player 1 when the game is on-going
P2	Refers to player 2 when the game is on going
Player Texture	A colored oval displayed under each player model.