CPSC 224 Final Project

PROJECT PLAN
<11/9/2024>

<Final Project Game: Snake>

<Binary Brawlers>

Prepared by:

Francesca Strickland-Anderson	**Cooper Braun**
<Put Student 1 Name Here>	<Put Student 2 Name Here>
Navin Kunakornvanich	**Bradley Russell**
<Put Student 3 Name Here>	<Put Student 4 Name Here>

1 Project Overview

1.1 Project Summary

We are making the game “Snake”. The player must keep the snake from colliding with both other obstacles and itself, which gets harder as the snake lengthens. The snake gets longer with each piece of food eaten. This is a single-player game. The game will require the snake, board, food, and scoreboard.

2 Project Requirements

2.1 Major Features

Major features needed for this game are the snake's movement and growth, detection if the snake collides with itself or a wall, food, display player score, welcome and game over message, difficulty settings (easy, medium, hard), and the board where the snake can move around. The goal is to make the game work!

Table 1: Major Features

Feature	Description
Snake Movement	Make sure the snake moves across the board smoothly in all directions.
Snake Growth	Have the snake grow when it eats food
Food	An item that causes the snake to grow. Apple +1 (grow) Rotten Apple -1 (shrink) Golden apple +5 (grow a lot) Star fruit - invulnerability
Collision	Detect when the snake collides with itself or a wall on the board. If there is a collision trigger the ‘game over’ screen
Score	Display accurate player scores throughout the game. The score is the amount of food eaten.
Welcome messages	Display a welcome message when the game starts. Prompt user to input their name and ask if they wish to play the game. Display a game over the screen that gets triggered by a collision. Display the player's score and ask if they wish to play again.
Difficulty	Three levels of difficulty: Easy, medium, and hard. Easy: No barriers Medium: Few barriers and increased moving speed Hard: Lots of obstacles and increased moving speed
Game board	The area where the snake moves around with walls/barriers that can cause collisions. Each difficulty will have a different themed board
Game over message	Display the ‘Game over’ message, player score, and time. Ask if the player wishes to try again
Win	Display the ‘Win’ message, player score, and time. Ask if the player wishes to

	play again
Pause screen	Pauses game

3 Project Game Design

3.1 Initial User Interface Design

Describe the general user interface layout, including a set of initial user interface design mock-ups. This can be done as a sketch if it's cleanly done by hand, or digitally using a drawing tool.

3.2 Initial Software Architecture

Backend Classes:

Game

- **Attributes**
  - GameBoard board - The game board where the snake moves
  - Snake snake - The main snake instance
  - Score score - Tracks the player's score
  - Timer timer (if we want a timer) - Tracks the game time
  - Difficulty difficulty - Sets the difficulty level
- **Methods**
  - startGame() - Initializes and starts a new game
  - pauseGame() - Pauses the game
  - resumeGame() - Resumes a paused game
  - endGame() - Ends the current game and triggers game-over actions
  - checkGameOver() - Checks if the game should end because of a collision or other conditions, like winning, etc.

Snake

- **Attributes**
  - List<SnakeSegment> body - List of segments representing the snake's body
  - Direction direction - The current direction of the snake's movement
- **Methods**
  - move() - Moves the snake in the current direction
  - grow() - Adds a new segment to the snake's body
  - checkCollision() - Checks if the snake has collided with itself or a wall
  - changeDirection(Direction newDirection) - Changes the snake's direction

SnakeSegment

- **Attributes**
  - Int x, y - Coordinates of the segment on the board

- **Methods**
    - Add a constructor to initialize the segment's position

Food (abstract)

- **Attributes**
    - Int x, y - Position of the food on the board
    - Int growthValue - Value that determines how much the snake grows or shrinks
- **Methods**
    - Abstract void applyEffect(Snake snake) - Defines the effect of the food on the snake

Apple, GoldenApple, RottenApple, StarFruit (subclasses of Food)

- **Attributes (inherited from Food)**
    - Add unique growthValue for each type
- **Methods**
    - applyEffect(Snake snake) - Implemented differently for each food type

GameBoard

- **Attributes**
    - Int width, height - Dimensions of the board
    - List<Obstacle> obstacles - List of obstacles on the board
    - Food food - Current food on the board
- **Methods**
    - spawnFood() - Places a new food item on the board
    - checkFoodEaten(Snake, snake) - Checks if the snake has eaten the food on the board
    - generateObstacles(Difficulty difficulty) - Generates obstacles based on the difficulty

Obstacle

- **Attributes**
    - Int x, y - Position of the obstacle on the board
    - Int width, height - Size of the obstacle
- **Methods**
    - Add a constructor to initialize the obstacle position and size

Score

- **Attributes**
    - Int points - The players current score
- **Methods**
    - updateScore(int increment) - Updates the score by the specified increment (eating food)
    - resetScore() - Resets the score to zero (either by death or win)

DifficultyManager

- **Attributes**
    - Int currentLevel - Tracks the current difficulty level (1 = easy, 2 = medium, 3 = hard)
    - Int snakeSpeed - Sets the speed of the snake, which increases with each level.

    - Int obstacleCount - Adjusts the number of obstacles as difficulty progresses
- **Methods**
    - advanceDifficulty() - Increases currentLevel and adjusts snakeSpeed and obstacleCount based on the level
    - resetDifficulty() - Resets the difficulty to easy (1) settings at the start of a new game

Timer (if we want)

- **Attributes**
    - Int startTime - Time when the game started
    - Int elapsedTime - Time since the game has started
- **Methods**
    - start() - Starts the timer
    - stop() - Stops the timer
    - getElapsedTime() - Returns the elapsed time

Note: Thought the timer thing looked cool in class but we don't really need it in our game. Up to you guys if we want to implement it or not

Frontend Classes:

GameWindow

- **Attributes**
    - JFrame frame - Main game window
    - GamePanel gamePanel - Panel where the game is displayed
    - ScorePanel scorePanel - Panel where the score is displayed
    - LevelPanel levelPanel - Panel displaying the current level (difficulty)
- **Methods**
    - initWindow() - Initializes and displays the game window.
    - updateDifficultyDisplay(int level) - Updates the displayed level information in LevelPanel

GamePanel

- **Attributes**
    - Snake snake - The snake to be displayed
    - Food food - Current food item on the board
    - List<Obstacles> obstacles - List the obstacles on the board
    - DifficultyManager difficultyManager - Manages the level progression and update the visuals
- **Methods**
    - paintComponent(Graphics g) - Draws the game components, including changes due to level progression
    - updateGame() - Updates the game state, redraws components, and checks if its time to increase difficulty

StartScreen

- **Attributes**
  - JLabel welcomeMessage - Displays the welcome text
  - JButton startButton - Button to start the game
  - JTextField playerNameField - Input for the player's name
- **Methods**
  - display() - Shows the start screen

GameOverScreen

- **Attributes**
  - JLabel gameOverMessage - Displays the game-over text
  - JLabel scoreDisplay - Shows the player's final score
  - JLabel levelDisplay - Shows the last level achieved
  - JButton playAgainButton - Button to restart the game
- **Methods**
  - display(int finalLevel) - Shows the game-over screen with the final level reached

PauseScreen

- **Attributes**
  - JLabel pauseMessage - Displays the pause text
  - JButton resumeButton - Button to resume the game
- **Methods**
  - display() - Shows the pause screen

ScorePanel

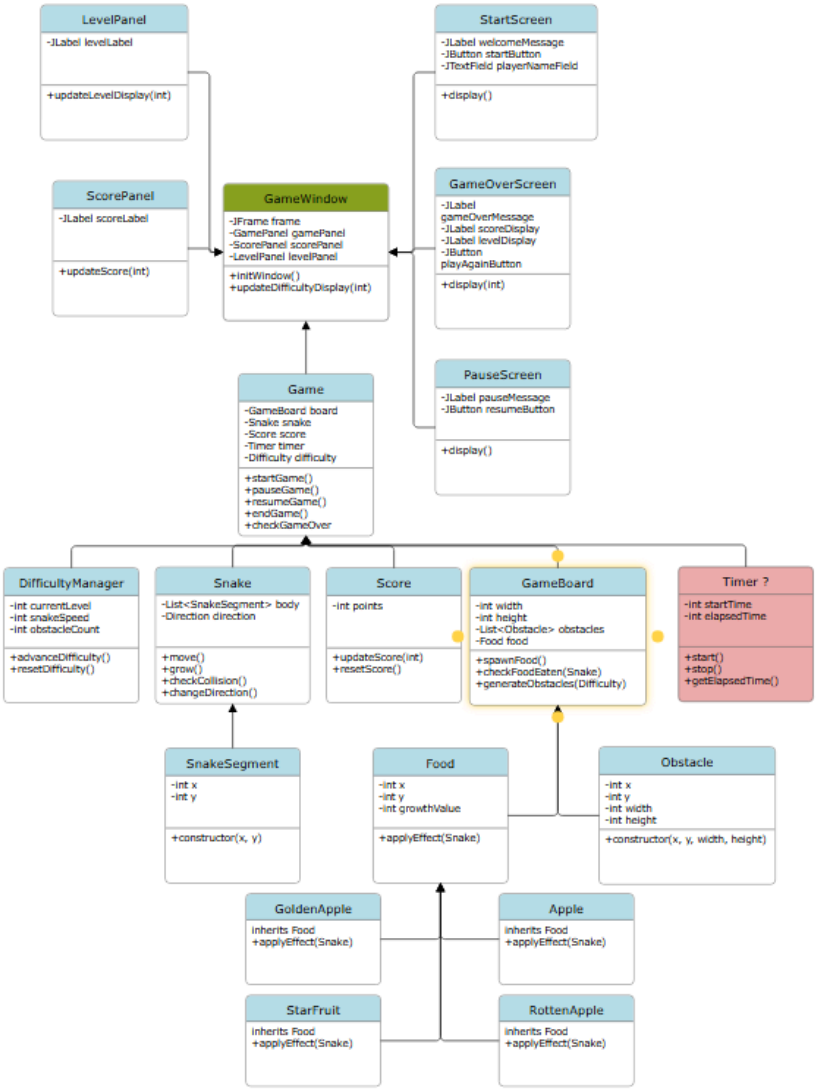
- **Attributes**
  - JLabel scoreLabel - Displays the current score
- **Methods**
  - updateScore(int score) - Updates the displayed score

LevelPanel

- **Attributes**
  - JLabvel levelLabel - Displays the current difficulty level ("Easy," "Medium," or "Hard")
- **Methods**
  - updateLevelDisplay(int currentLevel) - Updates the text in levelLabel based on the currentLevel (1 for "Easy," 2 for "Medium," 3 for "Hard")

4 Project Schedule

Describe the major scheduling dates of your project. For each scheduled milestone date, clearly describe the milestone (e.g., what features will be implemented) and when the milestone must occur. Include the project plan, code completion, presentation, and final report dates.


LevelPanel
-JLabel levelLabel
+updateLevelDisplay(int)
StartScreen
-JLabel welcomeMessage
-JButton startButton
-JTextField playerNameField
+display()
ScorePanel
-JLabel scoreLabel
+updateScore(int)
GameWindow
-JFrame frame
-GamePanel gamePanel
-ScorePanel scorePanel
-LevelPanel levelPanel
+initWindow()
+updateDifficultyDisplay(int)
GameOverScreen
-JLabel gameOverMessage
-JLabel scoreDisplay
-JLabel levelDisplay
-JButton playAgainButton
+display(int)
PauseScreen
-JLabel pauseMessage
-JButton resumeButton
+display()
Game
-GameBoard board
-Snake snake
-Score score
-Timer timer
-Difficulty difficulty
+startGame()
+pauseGame()
+resumeGame()
+endGame()
+checkGameOver
DifficultyManager
-int currentLevel
-int snakeSpeed
-int obstacleCount
+advanceDifficulty()
+resetDifficulty()
Snake
-List<SnakeSegment> body
-Direction direction
+move()
+grow()
+checkCollision()
+changeDirection()
Score
-int points
+updateScore(int)
+resetScore()
GameBoard
-int width
-int height
-List<Obstacle> obstacles
-Food food
+spawnFood()
+checkFoodEaten(Snake)
+generateObstacles(Difficulty)
Timer ?
-int startTime
-int elapsedTime
+start()
+stop()
+getElapsedTime()
SnakeSegment
-int x
-int y
+constructor(x, y)
Food
-int x
-int y
-int growthValue
+applyEffect(Snake)
Obstacle
-int x
-int y
-int width
-int height
+constructor(x, y, width, height)
GoldenApple
inherits Food
+applyEffect(Snake)
Apple
inherits Food
+applyEffect(Snake)
StarFruit
inherits Food
+applyEffect(Snake)
RottenApple
inherits Food
+applyEffect(Snake)

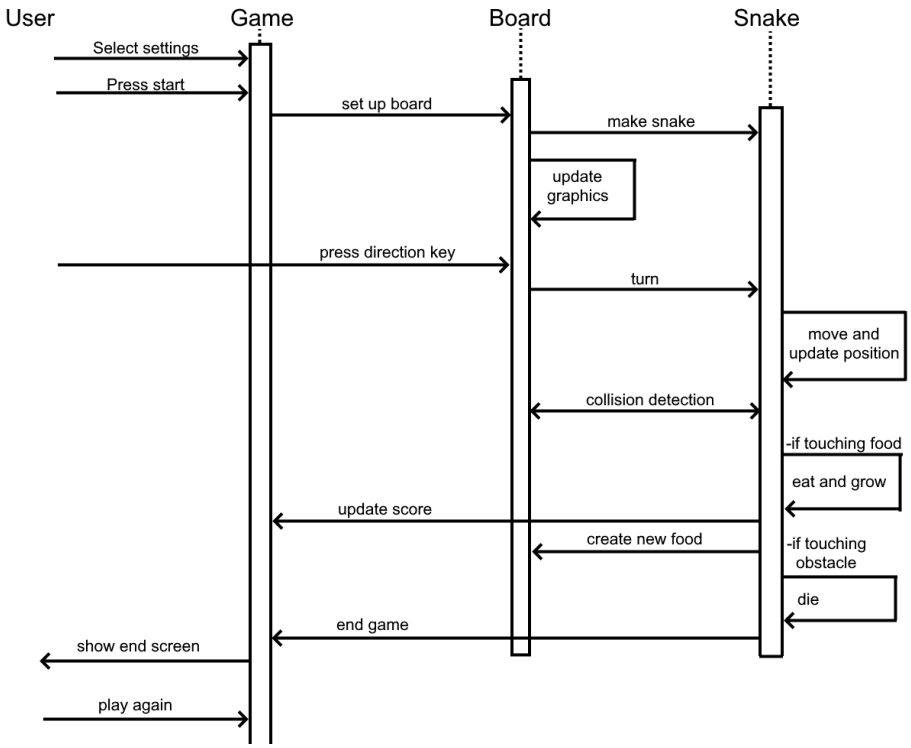


Table 3: Major Scheduling Milestones

Finish Plan & Design Interface	• Finish this doc • Create UML diagram, assign roles, class descriptions • Set up version control and create a shared repository • Assign roles for each member	Nov 9
Basic Game Mechanics, Collision Detection & Game Rules	• Implement game grid • Render initial snake and food • Basic game movements like left, right, up, down • Test to ensure the snake moves correctly • Add collision detection for walls and the snake's body to trigger game-over conditions. • Develop the food spawning logic and increase the snake's length with each food consumed. • Update the scoring system based on food eaten.	Nov 15
Level Implementation & Powerups	• Design each fruit/star with specific properties • Test the fruits, etc. to see if they’re balanced/work • Design the three levels with unique layouts and increased difficulty (obstacles, more bad fruit, etc.) • Test the levels are challenging but still fair and completable	Nov 22

Design Interface (Themes) & Test	● Create visuals for the snake, food, and different grids for levels ● Design basic start, pause, and end screen ● Design a score display ● Conduct playtesting sessions to catch bugs and improve user experience ● Create tests to make sure everything interacts correctly	Nov 29
Finish Up & Prepare Presentation	● Document the code and include comments for clarity ● Prepare a presentation on how each feature was implemented, and highlight any challenges and solutions ● Run a demo to show the game in action	Dec 6

Appendix

Provide additional supplemental information in an appendix as necessary.

As a developer, I want to be fun and interactive to capture the user's attention
As a player, I want to turn the snake by pressing arrow keys to move around the level.
As a player, I want to traverse the world by completing each level.
As a snake, I want to eat the food to grow long
As a snake, I want to avoid rotten food that will make me shrink
As a snake, I want to not run into myself or walls because then the round will end