1)OVERALL PLAN

Game Title: "Pixel Pioneers"

Game Concept:
In "Pixel Pioneers," players take on the role of an adventurous spaceship who must navigate a challenging space filled with asteroids(obstacles), enemies, and rewards. The objective is to guide the character from the starting point to the exit portal while collecting all regular rewards to achieve the highest score. However, players must be cautious, as moving enemies and punishments pose a constant threat. The game is set in a dynamic and visually engaging 2D environment.

Game Features:

- Main Character: Players control the main character using keyboard inputs. The character begins the game at the starting cell within the maze.

- Movement: The main character can move up, down, left, or right to adjacent cells, provided there are no barriers in the way. The character moves one cell per game "tick."

Collectible Rewards:

- Regular Rewards: The space is populated with regular rewards placed in various cells. Collecting these rewards adds to the player's score.
- Bonus Rewards: Occasionally, bonus rewards with higher point values appear randomly on the map. Collecting them enhances the player's final score. Bonus rewards disappear after a few ticks.

Barriers and Maze Design:

- Walls: The space is enclosed by forcefield on all four sides, with only two openings for the start portal and end portal.
- Maze Structure: The maze interior contains a series of blocked cells, creating a complex and challenging labyrinth.
- Individual Barriers: Additional barriers are placed on specific cells, preventing both the main character and enemies from passing through.

Enemies:

- Moving Enemies: There are animated moving enemies that actively pursue the main character. These enemies move one cell per tick, always aiming to get closer to the character.

- Punishments: Punishments (unanimated enemies) are placed on cells within the space. If the main character moves to a cell containing an asteroid(punishment), the player is penalized, potentially reducing their overall score or causing a loss.

Game Over Conditions:

- The player loses if the main character:
- Is caught by a moving enemy.
- Accumulates a negative overall score due to (hitting too many asteroids)punishments.
- The player wins by successfully navigating the maze, collecting all regular rewards, and reaching the exit portal.

User Interface:

- The game screen displays:
- The main character's current score.
- The time elapsed since the start of the game.
- The final score and time when the player wins.
- Clear and intuitive keyboard controls are provided for player interaction.

—----------

2)USE CASES

Use Case 1: Start New Game
Actor: Player
Description: This use case represents the player starting a new game.
Flow:

- Player launches the game application.
- The game presents the main menu.
- Player selects "Start New Game."
- The game generates a random space with (asterioids)obstacles, rewards, and enemies.
- The main character is placed at the starting cell.
- The game initializes the score and timer.
- Gameplay begins.

Use Case 2: Move Character
Actor: Player
Description: This use case represents the player moving the main character within the space.
Flow:

- Player presses the arrow keys (up, down, left, or right).
- The game checks if the move is valid (no barriers or forcefields in the way).
- If the move is valid, the main character moves in the selected direction.
- If the character moves to a space with a reward, it is collected and added to the player's score.
- If the character moves to a space with a punishment, the player's score may be reduced or the game may end.
- The game checks for game over conditions (collision with moving enemy, negative score, or win condition).
- Gameplay continues.

Use Case 3: Collect Bonus Reward
Actor: Player
Description: This use case represents the player collecting a bonus reward.
Flow:

- Occasionally, a bonus reward appears randomly on the map.
- The bonus reward is visible for a limited time (few ticks).
- If the main character moves to the cell containing the bonus reward during this time, it is collected.
- The bonus reward is removed from the map.
- The reward amount is added to the player's score.

Use Case 4: Encounter Moving Enemy
Actor: Player
Description: This use case represents the main character encountering a moving enemy.
Flow:

- Moving enemies pursue the main character in each game tick.
- If a moving enemy moves to the same cell as the main character, it results in a game over.
- The game displays a game over message.
- The game ends, and the player loses.

Use Case 5: Reach Exit and Win
Actor: Player
Description: This use case represents the player successfully reaching the exit and winning the game.
Flow:

- The player navigates the main character to the exit cell.
- The game checks if all regular rewards have been collected.
- If all regular rewards are collected and then exit is opened,
- If the player reaches the exit door, the player wins.
- The game displays a victory message.
- The final score and elapsed time are shown.
- The game ends, and the player wins.

—----------

3) UML DIAGRAMS

Explanation of Key Classes and Relationships for the UML diagram:

- Game: The central class that manages the game's state and logic. It contains references to other key components.

- Spacefield: Represents the space structure, including forcefields(walls), cells, and their layout. It provides methods for generating and rendering the maze.

- Character: An abstract class representing characters in the game, including the main character and enemies. It contains attributes such as position, score, and behaviors like movement.

- MainCharacter: A subclass of Character representing the player-controlled main character. It handles player input and interactions with rewards, punishments, and enemies.

- Enemy: An abstract class representing both moving and unanimated (punishment) enemies. It defines common enemy behaviors.

- MovingEnemy: A subclass of Enemy representing animated moving enemies. It handles their movement logic and AI for pursuing the main character.

- Reward: Represents rewards placed in the maze, including regular and bonus rewards. It has attributes like point value and a timer for bonus rewards.

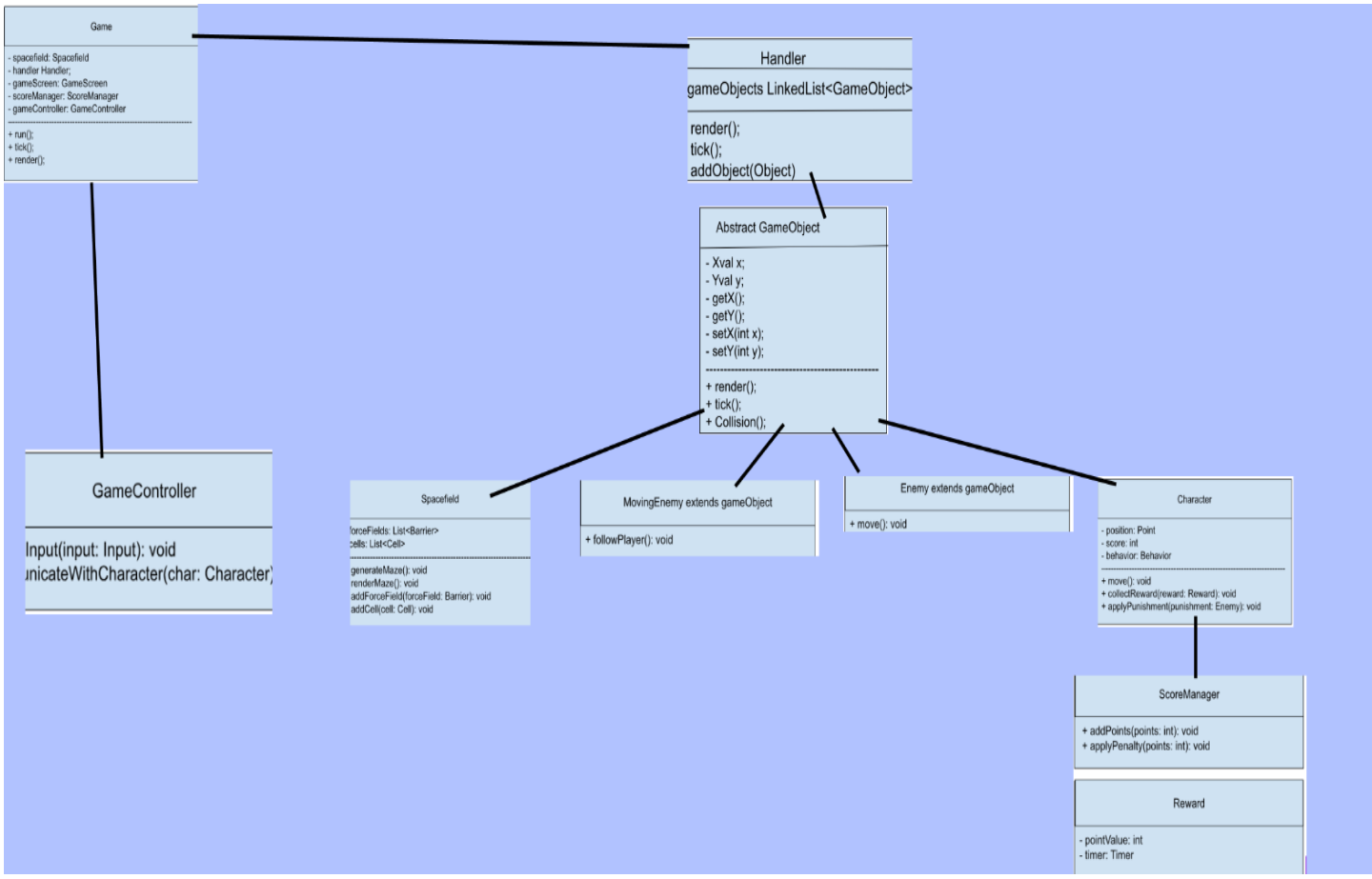
- Barrier: Represents barriers, including walls and individual barriers on specific cells. It prevents characters from moving through blocked cells.

- Timer: A utility class to keep track of the game's elapsed time.

- GameScreen: Handles rendering the game's graphics and user interface, including the character, maze, rewards, and score.

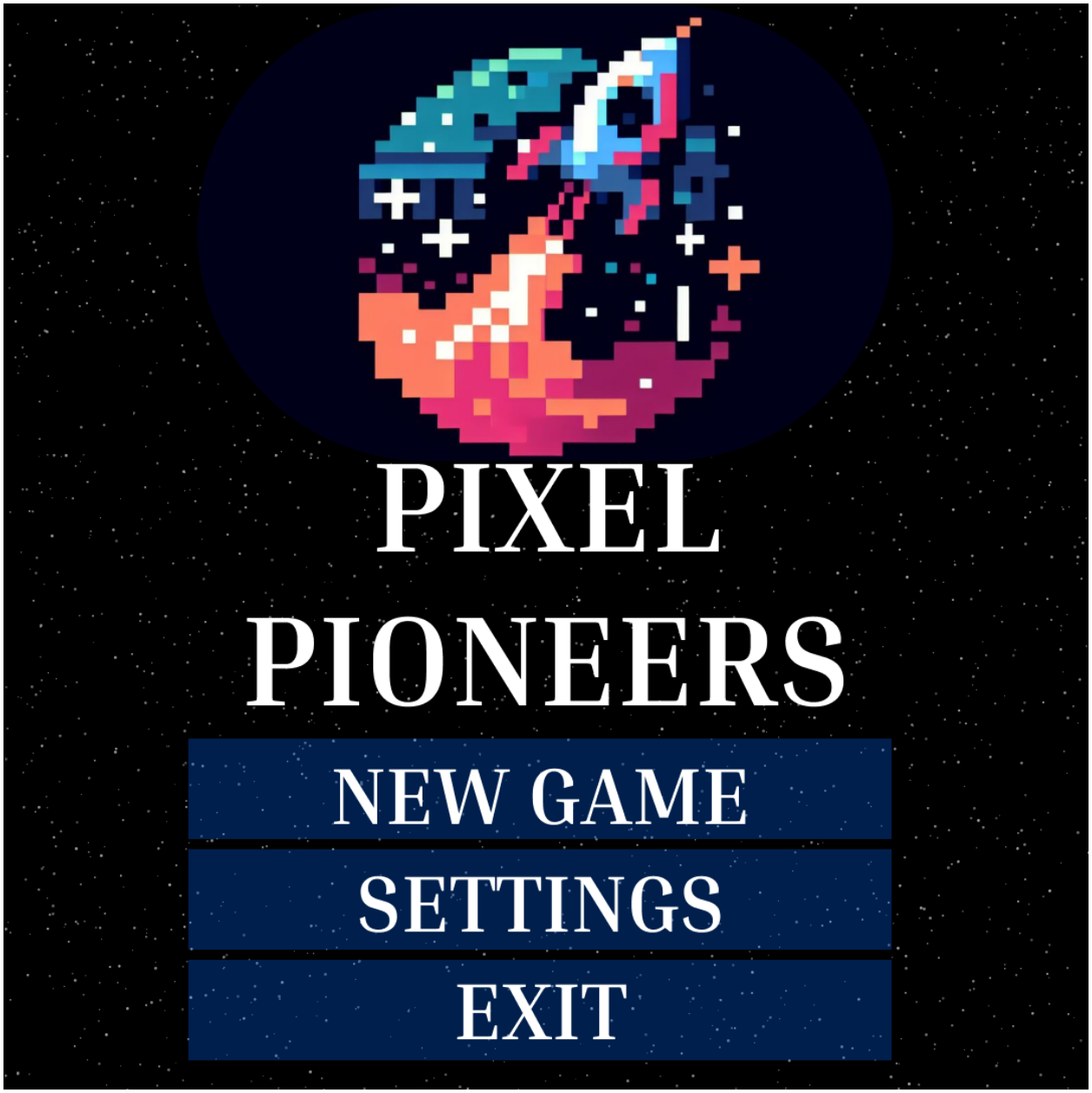
- ScoreManager: Manages the player's score, including adding points for collecting rewards and applying penalties for punishments.

- GameController: Handles user input, including keyboard controls, and communicates with the MainCharacter class to move the character.


Game
- spacefield: Spacefield
- handler Handler;
- gameScreen: GameScreen
- scoreManager: ScoreManager
- gameController: GameController
+ run();
+ tick();
+ render();
Handler
gameObjects LinkedList<GameObject>
render();
tick();
addObject(Object)
Abstract GameObject
- Xval x;
- Yval y;
- getX();
- getY();
- setX(int x);
- setY(int y);
+ render();
+ tick();
+ Collision();
GameController
Input(input: Input): void
unicateWithCharacter(char: Character)
Spacefield
forceFields: List<Barrier>
cells: List<Cell>
generateMaze(): void
renderMaze(): void
addForceField(forceField: Barrier): void
addCell(cell: Cell): void
MovingEnemy extends gameObject
+ followPlayer(): void
Enemy extends gameObject
+ move(): void
Character
- position: Point
- score: int
- behavior: Behavior
+ move(): void
+ collectReward(reward: Reward): void
+ applyPunishment(punishment: Enemy): void
ScoreManager
+ addPoints(points: int): void
+ applyPenalty(points: int): void
Reward
- pointValue: int
- timer: Timer

4) UI Designs

Main Menu UI



Game UI