IN2026 Games Technology

**Student Name:** Brian Poule

**Link to Git Repository:** https://github.com/brianpoule/IN2026-Games-Coursework

**Part I: Start Screen & Menu**

**Design and Implementation**

To create a more professional game experience, a start screen and menu system were implemented. This was done with a State machine.

* **Game State Management:** A GameState enum was added to Asteroids.h to define the possible states: STATE\_START\_SCREEN, STATE\_PLAYING, and STATE\_GAME\_OVER. A private member variable, mState, was added to the Asteroids class to track the current state.
* **GUI:** The game's GUI is now state-aware. On startup (STATE\_START\_SCREEN), a title and menu text are displayed using GUILabel objects, while the in-game UI (score and lives) is hidden by setting its visibility to false.
* **Transition to Gameplay:** A new private method, StartGame(), was created to handle the transition from the menu to gameplay. When the user presses the 'S' key on the start screen, this function is called. It changes the mState to STATE\_PLAYING, hides the menu UI, shows the in-game UI, and creates the spaceship and initial asteroids to begin the game.

**Key Code Snippets**

*The GameState enum from Asteroids.h:*

A black background with green text

AI-generated content may be incorrect.

*The logic in OnKeyPressed to start the game:*

A computer screen shot of a program code

AI-generated content may be incorrect.

**Screenshots**

**A screenshot of a video game

AI-generated content may be incorrect.**

**Part II: Bonuses & Power-Ups**

Three distinct power-ups were implemented to add variety to the gameplay. These power-ups are spawned periodically by a timer in the main Asteroids class.

**Power-Up 1: Extra Life**

* **Description:** An item a player could collect to increase their life by 1
* **Implementation:** A new ExtraLife class was created, inheriting from GameObject. When this object collides with the spaceship, it flags itself for removal. The Asteroids::OnObjectRemoved method detects when an "ExtraLife" object is removed, then gets the player's current lives from the Player object, increases it by 1, and updates both the Player object and the mLivesLabel.
* **Code Snippet (from Asteroids.cpp):**

A screen shot of a computer code

AI-generated content may be incorrect.

**Power-Up 2: Ship Teleportation**

* **Description:** An ability, activated by a key press 'T', that instantly teleports the spaceship to a random location on the screen. This is a high-risk, high-reward escape manoeuvre.
* **Implementation:** This feature was added directly to the Asteroids::OnKeyPressed method. When the 'T' key is pressed in the STATE\_PLAYING state, the function gets the screen dimensions from the mGameDisplay object. It then generates a random X and Y coordinate within these bounds and sets the spaceship's position to this new coordinate using its SetPosition method.
* **Code Snippet (from Asteroids.cpp):**

A screen shot of a computer program

AI-generated content may be incorrect.

* **Screenshot:** ship after teleporting

A screenshot of a computer screen

AI-generated content may be incorrect.

**Power-Up 3: Shield**

* **Description:** A collectable item that provides the spaceship with a one-time shield. The next time the ship collides with an asteroid, the shield is consumed, and the ship survives without losing a life.
* **Implementation:** A ShieldPowerUp class was created. A boolean flag mHasShield was added to the Spaceship class. When the ShieldPowerUp is collected, the Spaceship::ActivateShield() method is called, setting mHasShield to true. The Spaceship::OnCollision logic was modified to check for this flag. If true, it sets the flag to false and prevents the ship from being destroyed. If false, the ship is destroyed as normal.
* **Code Snippet (from Spaceship.cpp):**

A computer screen shot of text

AI-generated content may be incorrect.

**Part III: Version Control**

Version control was managed using Git, with a remote repository hosted on GitHub.

* A .gitignore file was configured to exclude temporary Visual Studio files (like the .vs folder) and build outputs from the repository.
* The development process followed a feature-based commit history. Major changes, such as the implementation of the start screen and each of the three power-ups, were saved in separate, detailed commits. This provides a clear and traceable history of the project's development.