CS/CE 224/272: Object Oriented Programming and Design Methodologies

Project Proposal: Color Switch Game

Team Members: Shifa Muhammad Imran, Hamna Munib, Sarah Faisal Alvi

November 8, 2025

Project Title: Color Switch			
Team Members' Details	Name	ID	Section
Team Member 1	Shifa Imran	sm10320	L6
Team Member 2	Hamna Munib	hm09929	L6
Team Member 3	Sarah Faisal Alvi	sa09646	L6

1 Introduction

We are building the popular mobile game, "Color Switch". The game will be built using core Object-Oriented Programming (OOP) concepts like classes, inheritance, and polymorphism to structure the game logic and components.

The game is a fast-paced, skill-based challenge where a player navigates an object through multi-colored obstacles, only passing through segments that match the object's current color.

The project will be implemented in C++ using the SFML (Simple and Fast Multimedia Library). This proposal details the game's core mechanics, the software design via a UML class diagram, and wireframes for key application screens.

2 Game / Application Details

2.1 Player

The player is represented by the **`Ball`** class, which inherits from a base `Element` class. It has attributes for `velocity`, `radius`, and `health`. Primary actions include `jump()` and `changeColor()`, along with collision detection methods.

2.2 Obstacles

Obstacles derive from an abstract **`Obstacle`** class. All obstacles have `Draw()`, `update()`, and `decreaseHealth()` methods. Concrete obstacle types include Paddle, Rotating Cross, and Box.

2.3 Collectibles

Interactive items inherit from the `Element` class:

- Star: Collectible with a `collect()` method and `points` attribute for scoring.
- ColorWheel: Features a `changeColor(Element*)` method to modify the ball's color.

2.4 Game Management

The **`Level`** class manages the game world, holding the `Ball`, `Score` object, and collections of `Element` and `Obstacle` pointers. The **`Score`** class tracks player progress with methods like `getScore()`, `incScore()`, and `decScore()`.

3 UML Diagram

The UML class diagram (Figure 1) presents our object-oriented design utilizing inheritance, composition, and aggregation relationships.

3.1 Core Architecture

Element Hierarchy: The abstract `Element` class serves as the base for all game objects with `position` (sf::Vector2f), `active` (bool), and `color` (sf::Color) attributes. It defines pure virtual methods `Draw()` and `update()` that must be implemented by derived classes.

Game Elements: Three classes inherit from `Element`:

- Ball: Player-controlled object with `velocity`, `radius`, and `health` attributes. Includes `jump()`, `changeColor()`, and `onCollision()` methods.
- Star: Collectible with `points` and `size` attributes, featuring a `collect()` method.
- ColorWheel: Power-up with `nextColor` and `radius` attributes, including `changeColor(Element*` method.

Obstacle System: The abstract `Obstacle` class provides `size` (Vector2f) attribute and methods `decreaseHealth()` and `increaseHealth()`. Three concrete obstacles inherit from this:

- Paddle: Simple obstacle with color matching
- Rotating Cross: Features rotation mechanics
- Box: Static obstacle with health modification capabilities

3.2 Game Management

Game Class: Manages overall game state with `state` (bool), `levels` (Level*), `TotalScore` (Score), `highestScore` (Score), and `player` (Ball). Methods include `startGame()`, `updateHighScore()`, and `getScore()`.

Level Class: Controls individual level gameplay, containing `score` (Score), `ball` (Ball), `elements` (Element*), and `obstacles` (Obstacle*). Provides `addElement()`, `addObstacle()`, `startLevel()`, and `endLevel()` methods.

Score Class: Tracks scoring with `score` (int) and methods `getScore()`, `incScore(Level &)`, and `decScore(Level&)`.

3.3 Key Relationships

- Inheritance: Ball, Star, ColorWheel inherit from Element; Paddle, Rotating Cross, Box inherit from Obstacle
- Composition: Game contains a collection of levels, 1 ball and 2 score(total score and high score)
- Association: Ball collides with Obstacles and collects Stars.
- Aggregation: Ball and score are part of each level but are retained even after the level ends.

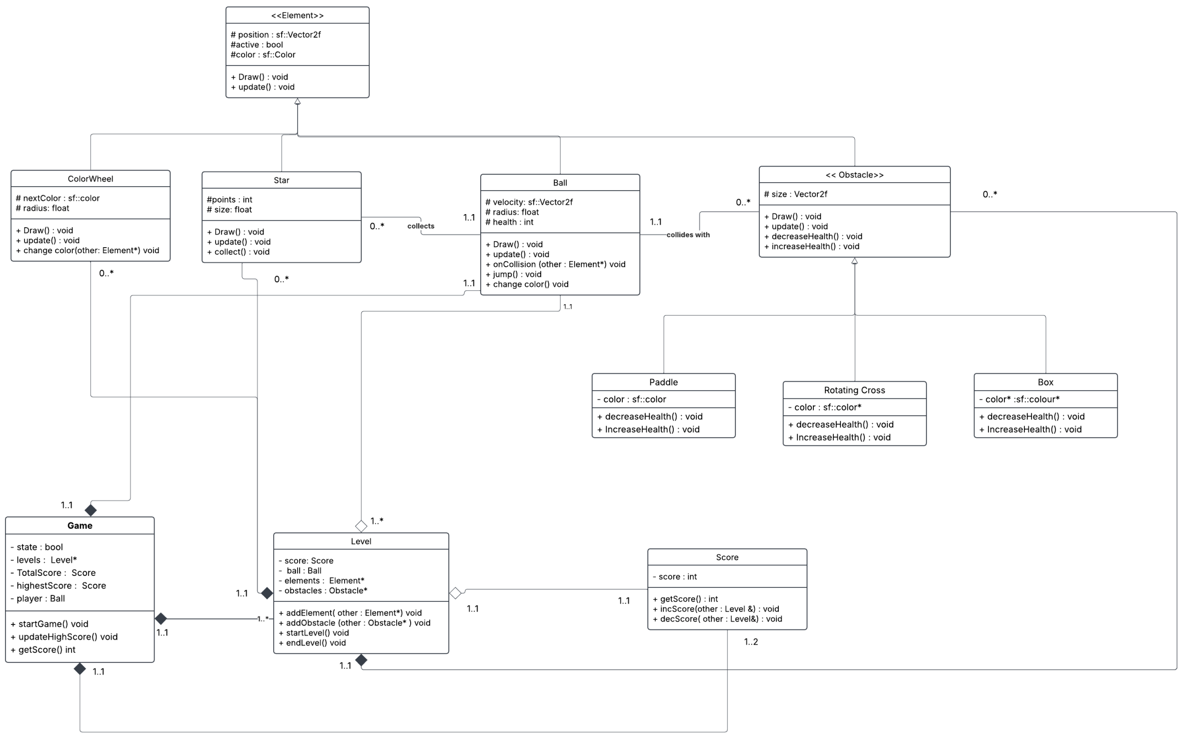

Figure 1: UML Class Diagram showing inheritance hierarchy and class interactions.

4 Game / Application Screens

4.1 Start Screen

The initial entry point displaying the “COLOR SWITCH” logo, start prompt, and decorative rotating `ColorWheel` elements.

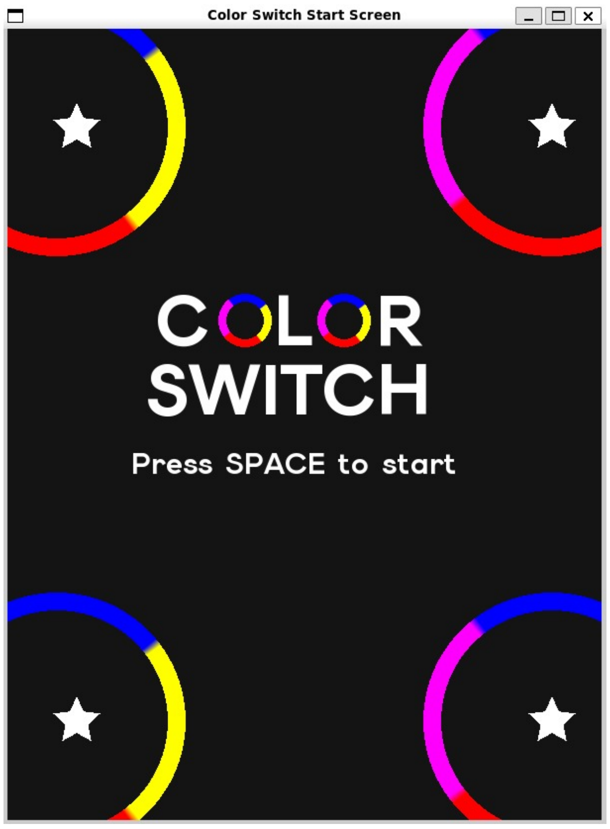

Figure 2: Start Screen Mockup.

4.2 Main Game Screen

Shows active gameplay with current score, pause button, player-controlled ball, and rotating obstacles.

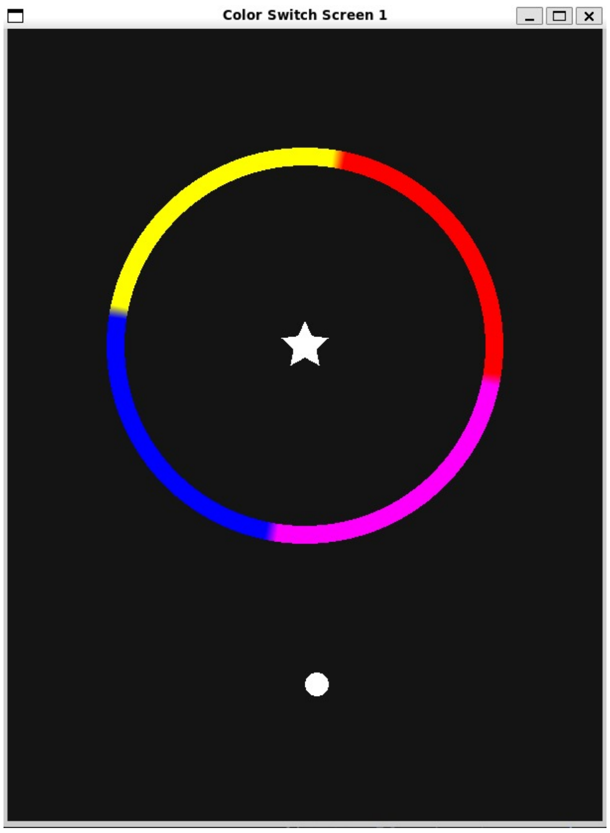

Figure 3: Main Game Screen Mockup.

4.3 Win Screen

Displays “LEVEL COMPLETE” message with score and options to proceed or return to menu.

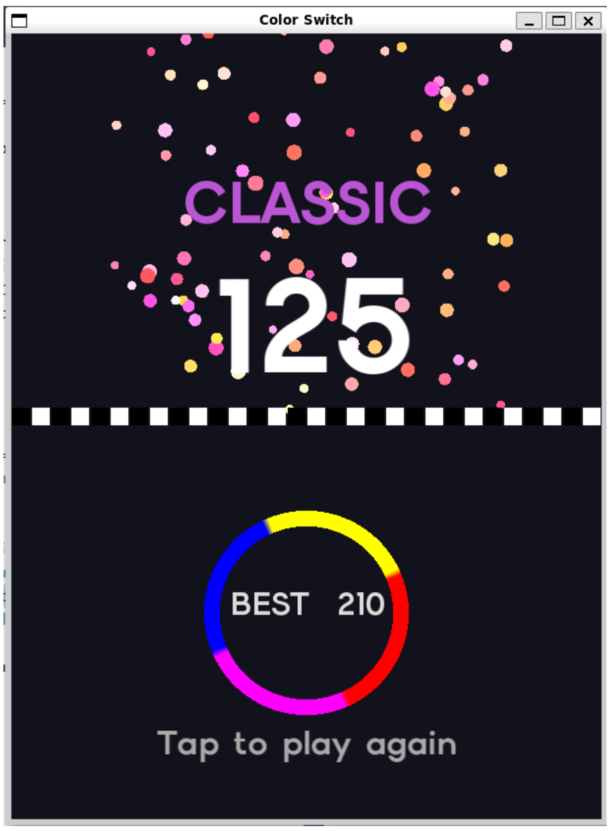

Figure 4: Win Screen Wireframe.

4.4 Score / End Screen

Game over screen showing final score, best score, and UI buttons for restart, high scores, and sharing.

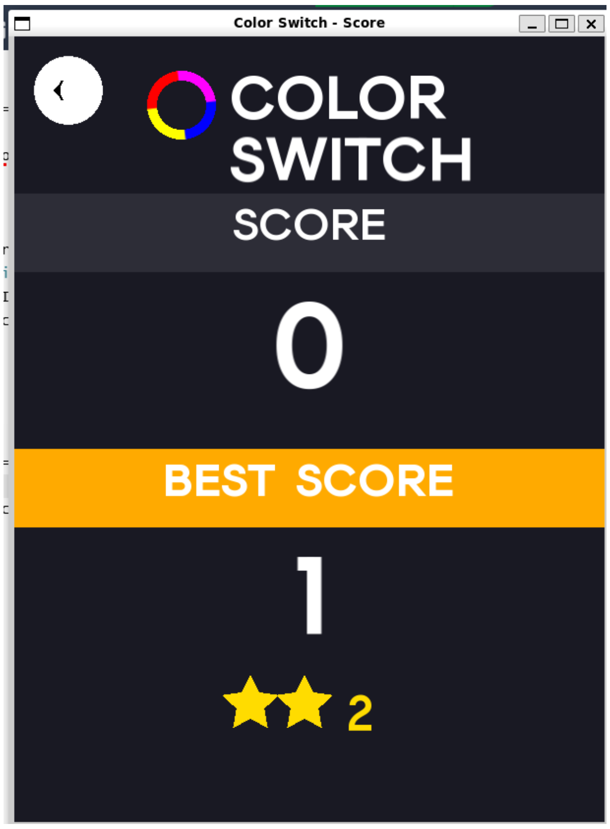

Figure 5: Score / End Screen Mockup.