```cpp
#ifndef GAME
#define GAME

#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>

// Make sure to add $(SFML_SDK)\include to the project properties under
// c++/General/Aditional include directories


/// <summary>
/// the four different modes our game can be in.
/// </summary>
enum class GameMode
{
	Showing,
	Recieving,
	GameOver,
	Starting
};




/// <summary>
/// @brief main class for the simon game.
///
/// this will be a single class simple game to mimic the 70 classic
/// game simon
/// </summary>
class Game
{
public:
	Game();
	void run();
protected:
	void update(sf::Time time);
	void startingUpdate();
	void recievingUpdate(sf::Time time);
	void showingUpdate();
	void overUpdate();
	void render();
	void processEvents();
	void setupButtons();
	void processGameEvents(sf::Event&);
	void resetButtons();
	void randomiseNotes();
	void countdownTimers();

	// colors for our buttons
	 const sf::Color RED{ 180,0,0,255 };
	 const sf::Color GREEN{ 0,180,0,255 };
	 const sf::Color BLUE{ 0,0,180,255 };
	 const sf::Color YELLOW{ 180,180,0,255 };
	 const sf::Color WHITE{ 255,255,255,255 };
	// main window
	sf::RenderWindow m_window;
	// red button / light
	sf::RectangleShape m_redSquare;
	// yellow button / light
	sf::RectangleShape m_yellowSquare;
	// blue button / light
```

```
sf::RectangleShape m_blueSquare;
// green button / light
sf::RectangleShape m_greenSquare;
// font used in the game
sf::Font m_impactFont;
// text for title
sf::Text m_titleText;
// text for blue instructions
sf::Text m_instructionsTextBlue;
// text for red instructions
sf::Text m_instructionsTextRed;
// text for Green instructions
sf::Text m_instructionsTextGreen;
// text for yellow instructions
sf::Text m_instructionsTextYellow;
// status text
sf::Text m_statusText;

// red buttone pressed
bool m_redButtonPressed;
// blue buttone pressed
bool m_blueButtonPressed;
// yellow buttone pressed
bool m_yellowButtonPressed;
// green buttone pressed
bool m_greenButtonPressed;

// time till blue button returns to normal color
int m_blueTimer;
// time till red button returns to normal color
int m_redTimer;
// time till green button returns to normal color
int m_greenTimer;
// time till yellow button returns to normal color
int m_yellowTimer;
// current time delay for buttons to return to normal color
int m_flashTime;
// delay between modes
int m_modeChangeTimer;

// sound buffer used to store tone wav
sf::SoundBuffer m_toneBuffer;
// sound for blue
sf::Sound m_blueTone;
// sound for red
sf::Sound m_redTone;
// sound for green
sf::Sound m_greenTone;
// sound for yellow
sf::Sound m_yellowTone;
// current mode the game is in
GameMode m_currentGameMdoe;
// array of notes
int m_noteSequence[32];
// difficulty target 8,16,32
int m_difficultyLevel;
// current note count
int m_currentNote;
// current sequence lenght
int m_currentCount;
// win state for gameover
bool m_win;
```

```
        // timer for turn taking
        sf::Time m_inputTime;
};

#endif // GAME
```