CS3A SFML PORTFOLIO

By: Sherman Yan

Department of Computer Science, Pasadena City College

Faculty Advisor: Dave Smith

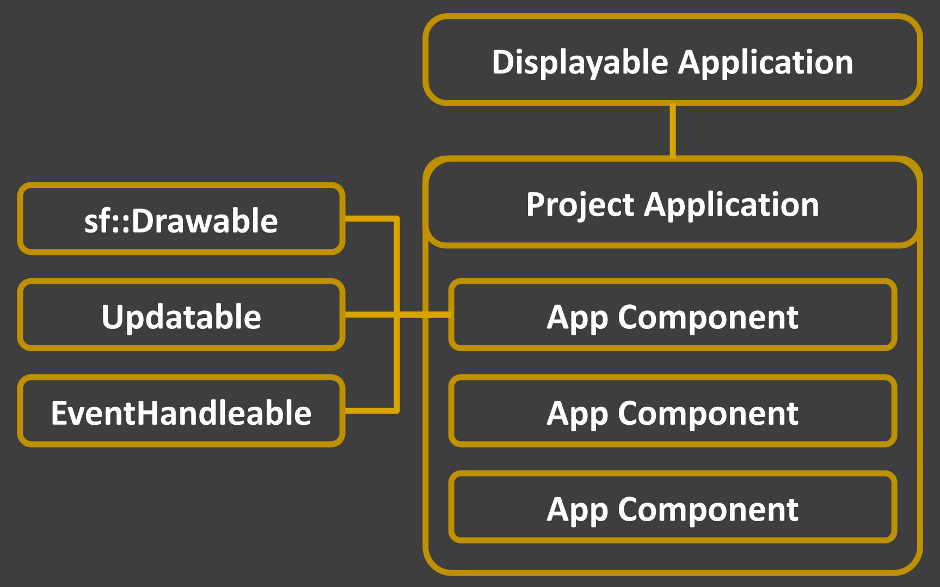
OVERVIEW

The SFML Portfolio is a capstone project covering all topics learned during the duration of the course with the utilization of SFML (A graphics library). The following are topics that are incorporated in this project:

- ➔ Object Oriented programming
- ➔ Inheritance
- ➔ Polymorphism
- ➔ Pointers
- ➔ SFML Graphics

STRUCTURE

The following is a diagram of how the application is structured. Each project inside the portfolio follows this format.



UTILITY CLASSES

DisplayableApplication

Utility class for creating a displayable application in SFML.

Public Member functions:

```
DisplayableApplication();
    Default constructor.
DisplayableApplication(const std::string& windowName);
    Construct window with window name.
DisplayableApplication(const std::string& windowName,
                       const sf::Color &bgColor);
    Construct window with name and background color.
void disableExit();
    Disable the Exit button.
void setWindowSize(const sf::Vector2u& windowSize);
    Set the window size.
void addComponent(AppComponent& component);
    Add component to display in window.
void run(sf::RenderWindow&window);
    Run application.
```

AppDriver

Static class to open apps.

Public Member function:

```
static void openApp(AppsEnum app,
                    sf::RenderWindow& window);
    Loads in the app in the given window..
```

Example Code:

```
switch (app) {
        case APP_OCEAN_CLEANUP: {
                App_OceanCleanup a;
                a.run(window);
                break;
        }
        case APP_POKER_ANALYSIS: {
                App_PokerAnalysis a;
                a.run(window);
                break;
        }
        case "rest of the apps…"
}
```

ScrollableContainer<T>

Utility class for creating a scrollable container in SFML.

Public Member functions:

```
ScrollableContainer();
    Default constructor.
ScrollableContainer(ScrollEnum scrollDirection,
                    float spacing);
    Construct container with scroll direction and spacing.
void scroll(float delta, const sf::FloatRect & bound);
    Scroll container delta amount between given bound.
void addComponent(T* item);
    Add the reference of component type T* the container.
void reverseScrollDirection();
    Reverse the scrolling direction.
void setItemSpacing(float spacing);
    Set the spacing between each item of the container.
sf::FloatRect getGlobalBounds() const;
    Get the container's global bounds.
void setPosition(float x, float y);
    Sets the top left position with coordinates x,y.
void setPosition(sf::Vector2f pos);
    Sets the top left coordinate with pos.
void update();
    Update the container item positions.
```

Example Code:

```
void scroll(float delta, const sf::FloatRect & bound){
    if (!items.empty()) {

        sf::FloatRect selfSize = getGlobalBounds();
        if (delta + selfSize.top <= bound.top &&
            delta + selfSize.top + selfSize.height >= bound.height)
                items[0]->move(0, delta * direction);
    }
}

void update(){
    if (!items.empty())

        for (int i = 1;i < items.size(); i++)
                Position::right(*items[i], *items[i - 1], spacing);
}
```

States

Utility base class to store object state.

Public Member functions:

```
States();
    Default constructor.
bool checkStates(StatesEnum state) const;
    Check if state is active.
void enableState(StatesEnum state);
    Enable state.
void disableState(StatesEnum state);
    Disable state.
void setState(StatesEnum state, bool value);
    Set state to value.
void toggleState(StatesEnum state);
    Toggle state.
```

Position

Static utility class for positioning objects.

Public Member functions:

```
template<class T, class S>
  static void left(T& self,const S& ref, float spacing =0);
      Align the left of self and reference object.
template<class T, class S>
  static void right(T& self,const S& ref, float spacing =0);
      Align the right of self and reference object.
template<class T, class S>
  static void top(T& self, const S& ref, float spacing =0);
      Align the bottom of self and reference object.
template<class T, class S>
  static void bottom(T& self, const S& ref, float spacing =0);
      Align the bottom of self and reference object.
template<class T, class S>
  static void center(T& self,const S& ref);
      Centers self with reference.
```

Fonts

Static utility class to get fonts.

Public Member functions:

```
static sf::Font& getFont(FontsEnum font);
    Returns the sf::Font based on font.
```

Textures

Static utility class to get Textures.

Public Member functions:

```
static sf::Texture& getTexture(TextureEnums texture);
    Returns the sf::Texture based on texture.
```

MouseEvents

Static utility class to check mouse events.

Public Member functions:

```
template<class T>
  static bool isHover(const T& Obj , const sf::RenderWindow& window);
      Checks if mouse is hovered over Obj.
template<class T>
  static bool isClick(const T& Obj , const sf::RenderWindow& window);
      Checks if Obj is clicked by mouse.
```

CONCLUSION

Reflection

This project allowed me to strengthen my understanding of all the different topics and tool that were taught during the duration of the course. As this was my first exposure to SFML and even creating UI graphics, this projects was eye-opening and allowed me to develop a good general understanding of graphics in games and other user interfaces as well as the structure of how games work.

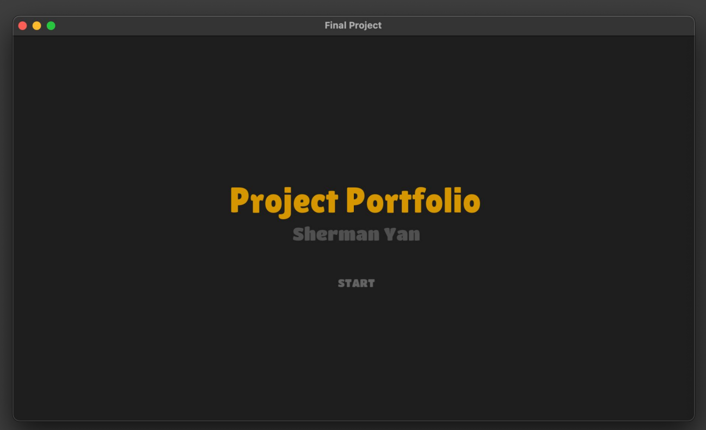
Next Steps

- ➔ Improve the DisplayableApplication class to make it more versatile.
- ➔ Fix DisplayableApplication class so that it can handle different interactions between different AppComponents to simplify the need for extra classes.
- ➔ Learn more advanced topics and tools to create new more advanced projects to add to portfolio.

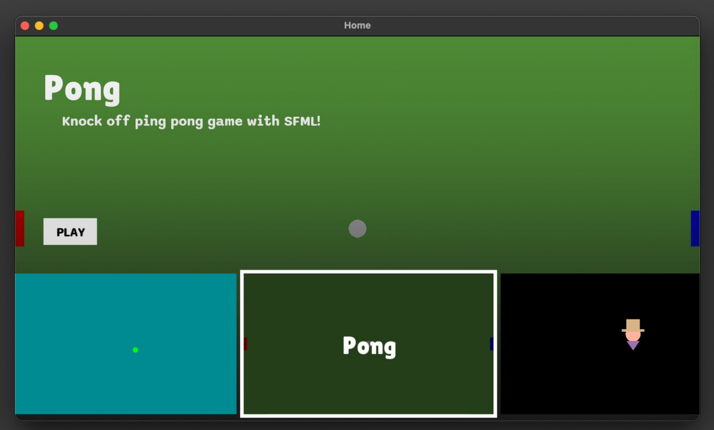
RESOURCES

<SFML/Graphics.hpp>
Author: SFML (Simple and Fast Multimedia Library)
Library version: SFML 2.5.1
Availability: https://www.sfml-dev.org/

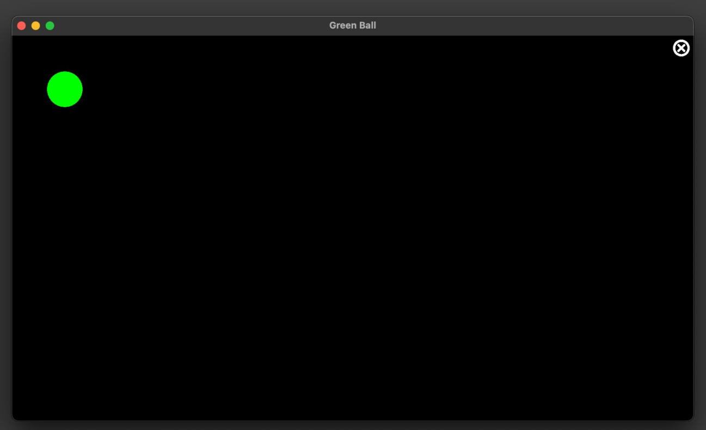
PROJECT RESULTS



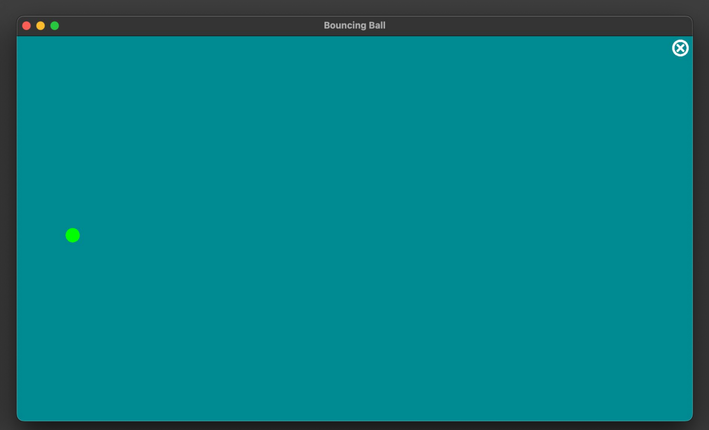
Splash Screen



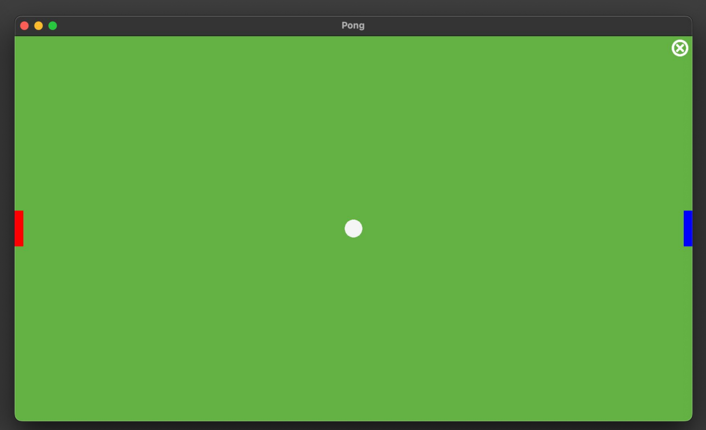
Gallery Display
Scrollable container to display all developed projects.



Green Ball
First SFML Project! Drawing a green ball.

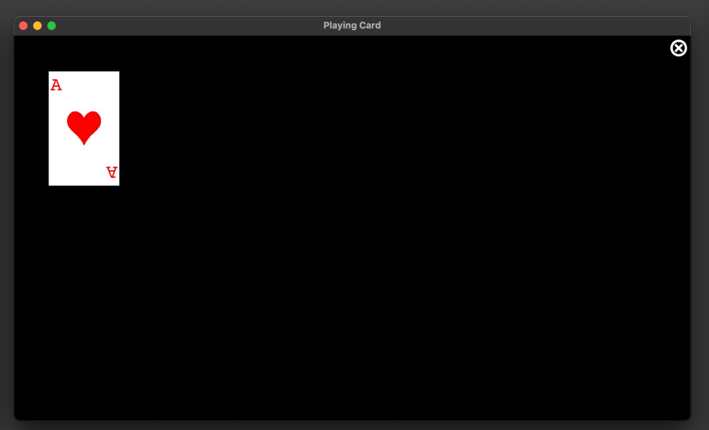


Bouncing Ball
Learning how to make object bounce inside a boundary.



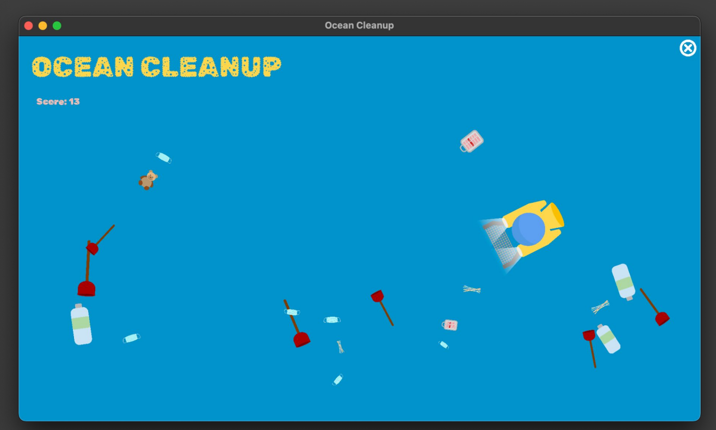
Pong
Knock off ping pong game with SFML!

Top Hat Guy
Creating drawable objects inheriting sf::Drawable.



Playing Card
Learning Sprites/Textures, and Text/Fonts.

Poker Analysis
Displaying Calculated Statistics with SFML.



Ocean Cleanup
Final Game! Collect trash and clean our ocean.