CS300/CS500
Advanced Object-Oriented Programming with C++

Lab #5:

Overview: In this week's lab you will be using the graphics library SFML to create a program that emulates the bouncing dvd logo. Using inheritance, you will develop three different screen savers (referred to as Savers) that put an interesting spin on the bouncing logo idea. You will use the abstract base class **ScreenSaver** as a parent class for all three of the Savers you will create.

We've provided a starting scaffold for the assignment, which includes:

- main.cpp: The file we will use to grade your assignment.
- screenSaver.h and screenSaver.cpp: Contains the abstract base class ScreenSaver.
  - You will add your other savers here.

Your task is to implement three child classes: **ClassicSaver**, **ColorChangingSaver**, and **CustomSaver**, and ensure they work with the provided main.cpp driver. If your implementation displays all savers with the expected behavior, you have successfully completed the assignment.

Finally, push your screenSaver.h and screenSaver.cpp files, along with a Makefile that builds the project, to a subdirectory in your CS300/500 GitHub repo called lab05. Then, submit the repo link on Canvas.

Part I: Create a class called **ClassicSaver** which more or less represents the classic dvd logo screensaver. It must inherit from **ScreenSaver** and have these features:

- A constructor called *ClassicSaver(float radius, sf::Vector2f velocity, sf::Color color)*
  - radius - defines the size of the radius of the sf::CircleShape
  - Velocity - defines the starting velocity of the saver
  - Color - defines the color of the saver
- Override the function update() from ScreenSaver
- The Saver Must move at a constant rate as defined by the by the initial velocity vector given to the constructor
  - The vector (1,0) would result in the saver moving right

    - The vector (1,1) would result in the saver moving at a 45 degree angle down and to the right
- Movement should be smooth and independent of the frame rate. To achieve this, scale the movement by deltaTime.
- The saver must bounce off the walls of the window. Use the provided **checkWallCollisions** function for this.
- The saver must move at a slow enough rate as to not look glitchy/nauseating

All of this functionality should be included in the update function that has been overridden from the **ScreenSaver** class. Don't overthink this! In my implementation the update function for this class has two lines: one to move the sf::CircleShape object, and one to check for wall collisions using the given function. Documentation **3 points**

Part II: Create a class called **ColorChangingSaver** which inherits from **ClassicSaver** and has the following features:

- A constructor called *ColorChangingSaver(float radius, sf::Vector2f velocity, float colorSpeed)*
    - radius - defines the size of the radius of the sf::CircleShape
    - Velocity - defines the starting velocity of the saver
    - colorSpeed - Determines how fast the color transitions occur. This is an arbitrary value, so whether 1.0f is fast or slow is up to your implementation
- Override the function update() from ScreenSaver
- Contains all features of **ClassicSaver**, including constant movement and wall bouncing.
- Smoothly transitions between colors as it moves across the screen.
    - The specific colors are flexible (e.g., transitioning between red and blue, green and purple, or cycling through the rainbow).
- The color transition is continuous and never-ending.
    - The saver should not settle on a single color after transitioning; it must constantly change colors throughout its movement.

Again all of this functionality should be included in the update function that has been overridden from the **ScreenSaver** class. Check this out to figure out how to work with colors. **3 points**

Part III: Create a class called **CustomSaver** which inherits from **ScreenSaver**. **CustomSaver** has no specific requirements, and is an opportunity for you to do whatever you want with this concept. You may choose to make a saver that moves around in a circular path, or teleports whenever it touches walls, maybe the saver changes size as it moves across the screen. Here are the Guidelines:

- **CustomSaver** must contain a constructor *CustomSaver(float someFloat, sf::Vector2f someVector, float someOtherFloat)*
    - Note that the names of these parameters and their purpose is completely up to you
- The Saver must be visible at all times
- The Saver should not create any flashing/glitchy imagery that is potentially dangerous for some to look at it.
- The Saver must be meaningfully different from **ClassicSaver** and **ColorChangingSaver**
- The Saver must have a unique gimmick or idea that is described in a comment above the class prototype in the screenSaver.h file
- The Saver must be totally cool and radical (just kidding but please make it interesting)

3 points

Part IV: Create a MakeFile with a default target that compiles the project, and a target called "run" that compiles then runs the project. If your Makefile is platform specific, or includes paths that are only relevant to your computer, that is fine.**1 point**

Part V: Push your screenSaver.h, screenSaver.cpp, and Makefile to a subdirectory called lab05 in your CS300/500 repo. There is no need to include a main.cpp file, and if one is included it will not be graded.