How to Create Your Own Game or Graphical Library

Welcome to the guide on how to create your own game or graphical library using the Arcade namespace! With Arcade, you have the power to create unique and engaging games that can be enjoyed by players all around the world. Whether you're an experienced developer or a newcomer to the world of programming, this guide will provide you with the information you need to get started.

Getting Started

Before you start creating your game or graphical library, you'll need to have a basic understanding of the Arcade namespace. Arcade provides a set of enums and structs that define colors, shapes, input actions, and more. Here's a quick overview of the most important enums and structs you'll be working with:

- `MouseEvent`: defines the types of mouse events that can occur, such as a right mouse button press or a left mouse button release.
- `Color`: defines the different colors that can be used to draw shapes and text.
- `Shape`: defines the types of shapes that can be drawn, such as rectangles, circles, and text.
- `InputAction`: defines the different types of input actions that can occur, such as pressing the up arrow key or the enter key.
- `Direction`: defines the different directions that shapes and sprites can face, such as north, south, east, and west.
- `MouseInput`: a struct that contains information about a mouse input event, such as the position of the mouse and the type of event that occurred.
- `BoardCell`: a struct that represents a cell on the game board. It contains information about the color, shape, size, and text of the cell, as well as the direction that any sprite on the cell is facing.

Creating Your Game

To create your game, you'll need to create a new class that implements the `IGameModule` interface. This interface provides a set of methods that your game must implement, such as `update`, `getBoard`, `onInputPressed`, and `getActualScore`. These methods are used to update the game state, draw the game board, handle input events, and retrieve the current score, respectively.

When implementing these methods, you'll have access to the `IDisplayModule` interface, which provides a set of methods for drawing shapes, text, and images on the screen, as well as getting user inputs and initializing and destroying the screen.

Creating Your Graphical Library

To create your graphical library, you'll need to create a new class that implements the `IDisplayModule` interface. This interface provides a set of methods that your graphical library must implement, such as `initScreen`, `destroyScreen`, `drawRectangle`, `drawCircle`, `drawText`, `cleanScreen`, `drawScreen`, `getInputPressed`, and `getMousePressed`. These methods are used to initialize and destroy the screen, draw shapes, text, and images on the screen, handle user inputs, and clean and draw the screen, respectively.

When implementing these methods, you'll have access to the `BoardCell` struct, which represents a cell on the game board. You'll use this struct to determine the color, shape, size, and text of each cell, and draw it on the screen using the appropriate methods.

Conclusion

With the Arcade namespace, creating your own game or graphical library is easier than ever before. By following the steps outlined in this guide, you'll be able to create engaging and immersive games and graphical libraries that will provide hours of entertainment for players all around the world. So what are you waiting for? Start creating today!

The Enums and Struct:

```
enum MouseEvent {
    rmbPressed = 0,
    rmbReleased,
    lmbPressed,
    lmbReleased
};

enum Color {
    BLACK = 0,
    WHITE,
    RED,
    BLUE,
    YELLOW,
    PURPLE,
    ORANGE,
    GREEN,
    CYAN,
    DARK_RED,
    DARK_BLUE,
    DARK_YELLOW,
    DARK_PURPLE,
    DARK_ORANGE,
    DARK_GREEN
};
enum Shape {
    RECTANGLE = 0,
    CIRCLE,
    TEXT,
    USERNAME,
    NONE
};
enum InputAction {
    UP = 0,
    DOWN,
    RIGHT,
    LEFT,
    BUTTON1,
    BUTTON2,
    BUTTON3,
    ENTER,
    PAUSE
};
```

```
enum Direction {
    NORTH,
    SOUTH,
    WEST,
    EAST
};

struct MouseInput {
    std::pair<int, int> mousePos;
    MouseEvent event;
};

struct BoardCell {
    Color shapeColor;
    Shape shape;
    std::pair<float, float> shapeSize;
    std::string textOrPath;
    Direction spriteDirection;
};
```