Pygame-ui

Pygame-UI is a component framework intended for rapid user interface development of games developed using the pygame library and for graphical user interface applications. Pygame-UI extends pygame by providing easily accessible and configurable commonly used components and tools. Pygame-UI also provides easily extendable base framework for development of custom components.

UI Component Framework for Python and Pygame

Vision

# Key Concept

Game development has several different aspects to it all of which contribute to the end-user experience. One of these aspects is the user interface (UI) development. Good UI is usually intuitive, easy to use, and visually appealing to the user. Pygame-UI provides extensible UI component framework and a toolbox of often used UI components. The main purpose of this framework is to reduce the time developers spend designing and implementing their user interface and thus to allow for more time for the development of game engines, game mechanics, etc. Pygame as a library does not provide any such user interface toolbox which requires from game developers to write their own using the provided pygame objects. This allows for developers to make custom components for each of their games, but some components are common to all such as buttons, labels, input boxes, menus, etc. Pygame-UI is intended to provide \customizable such components that are easily configurable and provide end-user experience equivalent to custom made.

Architecture

# Overall Design

Pygame-UI takes advantage of the pygame Surface object by wrapping it. The Surface object is then extended to abstract functionalities related to user interaction. Pygame-UI provides a component collection which handles user events in order for the components to respond to user interaction. The component collection also handles the rendering of all the components. Components that are not included in the component collection have to be manually provided the user events and their rendering method has to be manually called. Pygame-UI uses the pygame.events in order to queue up custom client events defined in the component framework. Thus when for example a button is clicked this would be reflected to the developer by a queued click event in the event queue and then the developer can respond to the user action appropriately. Pygame-UI also provides a configuration manager which handles XML definitions of components and provides an easy method of styling and declaring components in markup instead of programmatically.

Framework

# Key Classes

BaseUIComponent:

This is the base class for all components. This class provides all the components fundamental functionality, event handling methods and rendering method. Any custom component in the toolbox inherits from this class. For more information refer to the BaseUIComponent specification.

UIComponentCollection:

This class provides a wrapper for the component collection. The UIComponentCollection handles user interaction with the components and the rendering of the components. Once a component is added to the collection the developer has two methods he needs to call in his game loop: the Update method which takes as an argument a user event and updates the components according to the event and the Render method which re-renders all the visible components. For more information refer to the UIComponentCollection specification.

ConfigurationManager:

The configuration manager handles the parsing of XML markup that defines components. The key idea of this class is to define components non-programmatically thus speeding up the development time even more. The configuration manager initializes objects of type StylingManager, which is a class that handles the styling of the components. For more information refer to the ConfigurationManager specification.

ComponentStyling:

The component styling class allows for components to be styled using properties similar to web CSS styling. It parses a styling XML node defined for the specific component and it extracts properties defining positioning, colors, backgrounds, aligns, etc. StylingManager supports operations such as color conversion. For more information refer to the ConfigurationManager specification.

Toolbox

# Component List

TextLabel:

The TextLabel component is designed to display provided text with configurable size, font, color and hover actions. A single TextLabel can be reused to display different texts throughout your game/application by invoking its API to change the text, positioning and other configurations. For more information refer to the TextLabel specification.

ImageBox:

The ImageBox component is designed for loading and storing pygame supported image types. It supports three different states of the image: normal, hovered and clicked. For more information refer to the ImageBox specification.

Button:

The Button component is designed as a basic button component. It consists of an ImageBox defining the background of the button and a TextLabel holding the button’s text. The button click user event can be handled with a corresponding action or a custom callback can be given to the button to handle the event. For more information refer to the Button specification.

MessageBox:

The MessageBox component, which can also be referred to as a dialog component, is a pop-up message container. It is built using several already mentioned components and defines two parts: a general container holding the message TextLabel, and a header component holding a TexlLabel header and two Button components. The body holds the message text and can be minimized using the minimize header button. The header consists of a title, a minimize button which I’ve already mentioned and a close button which either destroys or hides the MessageBox. For more information refer to the MessageBox specification.

Future Agenda

# Generating Components from XML Markup

Currently the ConfigurationManager is used for defining the styling of the components, but this isn’t its only purpose. Essentially the configuration file would be able to define the entire component configuration such that there would be very little need for programmatic initialization because the framework would handle most of the work provided the configuration. This process would be described in more detail in the ConfigurationManager specification document.

# Full Support for CSS Styling Properties

Currently the StylingManager supports a subset of all CSS styling properties. The idea of this styling manager is to provide full support for all properties. Again more detail about this can be found inside the ConfigurationManager specification document.

# Designer and Design Time

The grand idea behind the ConfigurationManager is to ultimately serve as the base of a Pygame-UI Designer application. At this stage it is a bit early to speak of the designer as not nearly enough toolbox components have been developed in order to be used for such a designer application to be developed. This would still remain the core vision point for the ConfigurationManager and more information can be found inside the ConfigurationManager specification document.