P532 OOSD - Fall 2021

Team 3: Isaiah Sherfick, Snehal Patare, Raghu Gattu, Satin Patare

Week 4/5: GameMaker

Design Document

We believe our game maker project has a future-proof and scalable architecture that is likely the most impressive thing any of us have made. It utilizes two different MVC contexts to accomplish both game creation and play for the user. It uses an extremely powerful, extensible, and flexible Sprite class to represent all objects in its games. We took advantage of the composite, observer, strategy, null object, command, and MVC patterns to complete the assignment.

The true power of our system lies in the Sprite class at the heart of our model. Sprites contain two very powerful data structures: CustomCollisionMap and EventStrategyLinkedList. The first is used to enable the user to create a custom relationship between any two arbitrary sprites in the system and set a custom collision behavior for when a collision occurs between those two sprites. This is extremely liberating for the user. A ball in a Breakout game could bounce off one brick, yet be destroyed by another if the user so desired; there is no restriction on the “type” of Sprite because that concept does not exist.

The second data structure that enables our Sprites to do it all is the EventStrategyLinkedList. Our Sprites utilize a simple EventStrategy interface which consists of onclick, onkeypress, and onclocktick behaviors. However, Sprites are not limited to containing only one EventStrategy -- that would mean a new EventStrategy-compliant class for basically every behavior more complex than “move automatically in one direction.” Sprites contain a Linked List of these strategies, and that allows the user to not only chain different simple behaviors together to create a more complex one, but also to decide what order they want the behaviors to execute in on an atomic level.

The true beauty of the EventStrategy implementation in our system is that most simple games which our system cannot currently create can be accomplished by simply injecting one or two new EventStrategy classes into the system.

The CreateGameController utilizes the command pattern and serves as a bridge between the CGV and CGM. Each modification the user requests is encapsulated and stored for potential undo requests. The PlayGameController contains an observable clock which runs in its own thread, and it is responsible for being the “game engine” for the play context. The CreateGameView has an interface inspired by the GNU Image Manipulation Program with three separate windows that the user can resize and rearrange. The PlayGameView is inspired by our previous Breakout interfaces but with an added twist of a separate window for meta buttons.