# CS/SE -6387 Advanced Software Engineering Project

**Summer 2013**

Table of Contents

[Introduction 3](#_Toc361611738)

[System Description 3](#_Toc361611739)

[Architecture Style 4](#_Toc361611740)

[UML Diagrams 7](#_Toc361611743)

[Class Diagram 7](#_Toc361611744)

[Sequence Diagrams 7](#_Toc361611745)

[References 8](#_Toc361611746)

# Introduction

As Albert Einstein quoted - “Play is the highest form of research”. When we are playing, we are open to all possibilities and the acceptance to learn new things is much quicker. Games have a reputation for being fun and engaging; more importantly immersive, requiring deep thinking and complex problem solving. We believe educational games are essential in the next generation of e-learning tools. An extensible, freely available, engaging, problem-based game platform that provides students with an interactive simulated experience closely resembling the activities performed in a (real) industry development project would transform the SE/Sys education infrastructure.

Our goal is to create a Game Play Engine, which is dynamic and also cater to the needs of the education system that uses the game play engine for learning and testing purposes. Consider a scenario, where the students studying a chapter , for example “Calculus”, the students can learn a lot easier, if the assignments and exams are in the form of game, such as gaining points and moving up a level, upon gaining certain points, etc. So, the game play engine can help students grasp even the tough concepts very quickly.

# System Description

SimSys, a game play engine is being developed with the passive model of MVC (Model-View-Controller) architecture pattern. This system enables the user to provide the game in the form of XML document and that would be used to create a game, accordingly. The architecture that was used is a passive model of MVC pattern and the model is also a finite state machine. So, whenever there is a change or update in the model, all the views that are registered to a model will be notified of the change. Further information will be provided in the Architecture and Design sections of the paper.

Each game has a purpose (education topic or domain) and one or more learning objectives. A game has one or more characters. A game has one or more acts. Each act has one or more scenes. Each scene has one or more screens. Each scene has one or more game elements.

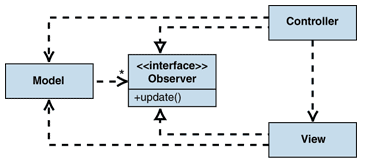
There are also transitions, which is basically a cut from an act to act or from scene to scene or from screen to screen. The system is developed using Java and Java Swing and additional libraries such as JAXB is used for parsing XML and converting into XML object.

# Architecture Style

# The architecture used for the project is Model-View-Controller pattern with passive model. The MVC architectural style is a software architecture pattern which emphasizes on the separation of information from user’s interaction. The key advantage of MVC architecture style is code reusability and separation of concerns.

# The Model-View-Controller (MVC) pattern separates the modeling of the domain, the presentation, and the actions based on user input into three separate classes:

1. **Model:** The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). Model on the whole, encapsulates the application’s state, responds to state queries, notifies views of its changes and provides the functionality of the application.
2. **View:** The view manages the display of information. View renders the model, requests updates from the model. View also allows the controller to select the view and it sends the user’s actions to the controller, which then sends it on to the model.
3. **Controller:** The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate. Controller actually defines the application behavior and also maps the user actions onto the model updates.



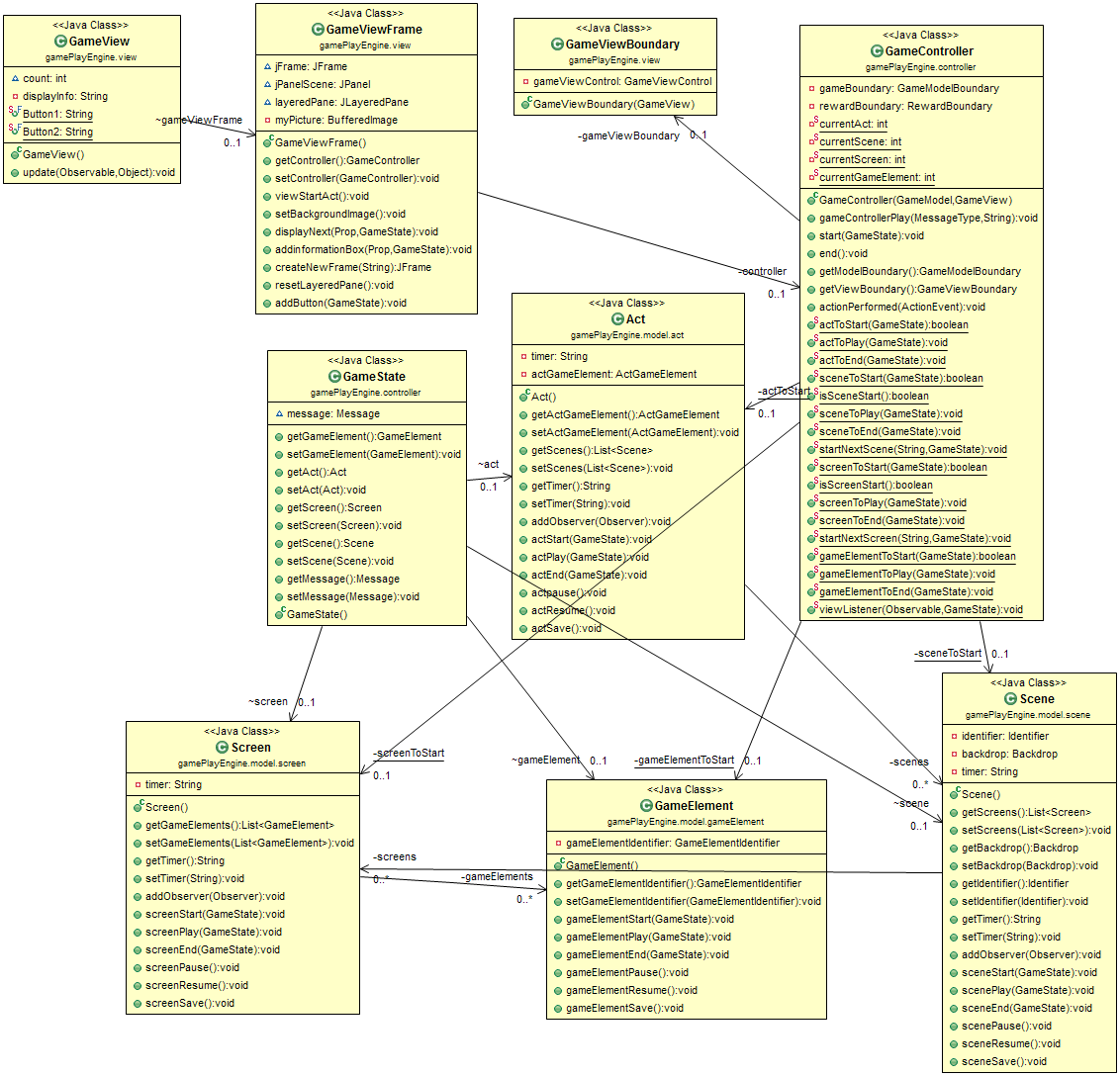
In the passive model of MVC pattern, the individual views implement the Observerinterface and each view registers with the model. The model tracks the list of all observers that are registered to it. When a model changes, the model iterates through all registered observers and notifies them of the change. This approach is often called "publish-subscribe." The model does not need specific information about views. In fact, in scenarios where the controller needs to be informed of model changes (for example, to enable or disable menu options), all the controller has to do is implement theObserver interface and subscribe to the model changes. When there are many views, multiple subjects could be defined. Each of these subjects would describe a specific type of model change. Each view can then subscribe only to types of changes that are relevant to the view.

Also the view must notify the model, if the update pushed to the model is complete. This is a place where, the controller comes to play. The logic that is specific to the application is present in the controller and not in the model. By keeping the logic in the controller, the views and models can remain independent of each other; there by reuse is one of the innate qualities of such architecture. In software engineering, reuse is one of the key benefits, instead of developing a module or code from the scratch.

In our project, we have used the passive model of MVC architecture style. When discussing about MVC architecture, there are two ways by which the data is exchanged between view and controller. It is either a pull or a push to send and receive data between view and controller. According to the push model, the user actions should be understood by the controller and then the controller would generate data and push it on to the view. So, this is called the push model. According to the pull model, when the view needs some type of information, the view would access the controller for getting the data it requires in order to display something based on the user’s request. This is the pull model, in which the view pulls the data from the controller.

# UML Diagrams

## Class Diagram



## Sequence Diagrams

# References

1. The Object Aid UML Explorer for Eclipse. [Online]

<http://www.objectaid.com/>