Table of Contents

[1.1 Abstract 3](#_Toc364251192)

[1.2 List of Figures 3](#_Toc364251193)

[1.3 Introduction 3](#_Toc364251194)

[1.4 Game Play Engine Requirements 3](#_Toc364251195)

[Test Game 1 Requirements 3](#_Toc364251196)

[Test Game 2 Requirements 8](#_Toc364251197)

[Test Game 3 Requirements 14](#_Toc364251198)

[Test Game 4 Requirements 26](#_Toc364251199)

[1.5 Game Play Engine Architecture 36](#_Toc364251200)

[1.5.1 Style 36](#_Toc364251201)

[1.5.2 Structural View (Component Diagram) 37](#_Toc364251202)

[1.5.3 Behavioural View (Sequence Diagram) 37](#_Toc364251203)

[1.6 Game Play Engine Design 41](#_Toc364251204)

[1.6.1 Structural View (Class Diagram) 41](#_Toc364251205)

[1.6.2 Behavioural Views (Sequence Diagram) 44](#_Toc364251206)

[1.7 Code 55](#_Toc364251207)

[1.7.1 Coding Standards, Sample Code, Libraries Used 55](#_Toc364251208)

[1.8 Testing 58](#_Toc364251209)

[1.8.1 Testing Approach 58](#_Toc364251210)

[1.8.2 Test Game 4 Sample Test Case 58](#_Toc364251211)

[1.9 References 64](#_Toc364251212)

CS6V81-571-ADVANCED SOFTWARE ENGINEERING PROJECT

SUMMER 2013

Guided By

Professor Dr. Kendra M.L. Cooper

Associate Professor

Department of Computer Science

Erik Jonsson School of Engineering & Computer Science

INIYAN PARAMASIVAM

([ixp110020@utdallas.edu](mailto:ixp110020@utdallas.edu))

SUDHARSAN LAKSHMINARASIMHAN

([sxl109520@utdallas.edu](mailto:sxl109520@utdallas.edu))

## Abstract

Software Engineering (SE) and Systems Engineering (Sys) are knowledge intensive, specialized, rapidly changing disciplines; their educational infrastructure faces significant challenges including the need to rapidly, widely, and cost effectively introduce new or revised course material; encourage the broad participation of students; address changing student motivations and attitudes; support undergraduate, graduate and lifelong learning; and incorporate the skills needed by industry. 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.

## List of Figures

## Introduction

Software Engineering (SE) and Systems Engineering (Sys) are knowledge intensive, specialized, rapidly changing disciplines; their educational infrastructure faces significant challenges including the need to rapidly, widely, and cost effectively introduce new or revised course material; encourage the broad participation of students; address changing student motivations and attitudes; support undergraduate, graduate and lifelong learning; and incorporate the skills needed by industry. 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.

## Game Play Engine Requirements

## Test Game 1 Requirements

Table 1 Act 1

|  |  |  |
| --- | --- | --- |
| Identifier | Act 1. | |
| Purpose | Start, Play, and End Act 1. | |
| Learning objectives | Not present. | |
| Uses Scenes | Scene 1 Game Welcome.  Straight Cut into, out of Scene 1. | |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Act | |
| Alternate flow of events | | If an error occurs, then end the game. |

Table 2 Act 1, Scene 1 Game Welcome

|  |  |  |
| --- | --- | --- |
| Identifier | Act 1. | |
| Purpose | Start, Play, and End Act 1. | |
| Learning objectives | Not present. | |
| Uses Scenes | Scene 1 Game Welcome.  Straight Cut into, out of Scene 1. | |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Act | |
| Alternate flow of events | | If an error occurs, then end the game. |

Table 3 Act 1, Scene 1, Screen 1 Play to Win Button Screen

|  |  |  |
| --- | --- | --- |
| Identifier | Screen 1. |  |
| Purpose | Present a play to win button to the player, which awards points. |  |
| Learning Objectives | Not present. |  |
| Declarations, initialization | | |
| Challenge | Not present. |  |
| Characters | | |
| Player | Default Player. |  |
| Non-player characters | Not present. |  |
| Setting (visual) | | |
| Props | | |
| Generic interaction | |  |  |  | | --- | --- | --- | |  | Button 1  Text: Play to Win!  Size: MEDIUM  Location: UC |  | | |
| Set Decorations | Not present. | |
| Audio | | |
| Music | Not present. |  |
| Challenge | Not present. |  |
| Game Play | | |
| Start of Screen | 1) Backdrop BlueSky.  (defined for the scene) |  |
| Interactions  (normal flow of events) | **Note. A button is presented to the player. When the player clicks it, they are rewarded with 5000 points.**  Start Screen 1  Play Screen 1  FADE IN the Button 1 as a QUICK EFFECT  When the player clicks the button, they are rewarded with  5000 additional points.  FADE OUT the Button 1 as a QUICK EFFECT  End Screen 1  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | Play to Win! |
| Alternate flow of events | If an error occurs, then end the game. | |
| End of Screen |  |  |

Table 1 Act 1 Scene 1 Screen 2 You have Won Screen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Screen 2 | |  | |
| Purpose | Present feedback about their points and a button to end the game. | |  | |
| Learning Objectives | Not present. | |  | |
| Declarations, initialization | | | | |
| Challenge | Not present. | |  | |
| Characters | | | | |
| Player | Default Player. | |  | |
| Non-player characters | Not present. | |  | |
| Setting (visual) | | | | |
| Props | | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Bubble 1  Text: You Won!!! Your current points are <points>!  Size: MEDIUM  Location: UC | Button 1  Text: End Game  Size: MEDIUM  Location: DLC |  | | | | |
| Set Decorations | Not present. | | | |
| Audio | | | | |
| Music | Not present. | | |  |
| Challenge | Not present. | | |  |
| Game Play | | | | |
| Start of Screen | 1) Backdrop BlueSky  (defined for the scene) |  | | |
| Interactions  (normal flow of events) | **Note. The feedback to the player is presented; it includes the updated number of points. A button to end the game is presented.**  Start Screen 2  Play Screen 2  FADE IN the Information Bubble 1 as a QUICK EFFECT  FADE IN the Button 1 as a QUICK EFFECT  Player clicks Button 1 to end the game.  FADE OUT the Button 1 as a QUICK EFFECT  FADE OUT the Information Bubble 1 as a QUICK EFFECT  End Screen 2  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | End Game  You Won!!! Your current points are 6000! | | |
| Alternate flow of events | If an error occurs, then end the game | | | |
| End of Screen |  |  | | |

## Test Game 2 Requirements

Table1 Act 1

|  |  |
| --- | --- |
| Identifier | Act 1 |
| Purpose | Start, Play, and End Act 1. |
| Learning objectives | Not present. |
| Uses Scenes | Scene 1  Scene 2  Scene 3  Straight Cut into, out of Scene 1, Scene 2, Scene 3. |
| Behaviour  state machine description:  Current Scene  Transition  event  condition  output  Next Scene | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Act |

**Table 2.2 Act 1, Scene 1**

|  |  |
| --- | --- |
| Identifier | Scene 1 |
| Purpose | Start, Play, and End Screen 1, Screen 2, Screen 3 with a White backdrop. |
| Learning objectives | Not present. |
| Uses Screens | Screen 1  Screen 2  Screen 3  Straight Cut into, out of Screen 1, Screen 2, and Screen 3. |
| Backdrop | White |
| Behaviour  state machine description:  Current screen  Transition  event  condition  output  Next screen | Initial state for Scene 1  START SCENE 1  PLAY SCENE 1  END SCENE 1  Final state for Scene 1 |
| Alternate flow of events | If an error occurs, then end the game. |

Table 3 Act 1, Scene 1, Screen 1

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 1 |  | |
| Purpose | Present an information box with LightBlue color |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: none  Size: MEDIUM  Color: LightBlue  Location: C |  |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1) Backdrop White  (defined for the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box is presented to the player; the information box fades in and then fades out.**  Start Screen 1  Play Screen 1  FADE IN Information Box 1 as a QUICK EFFECT  FADE OUT Information Box 1 as a QUICK EFFECT  End Screen 1  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | |  |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

**Table 4 Act 1 Scene 1 Screen 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 2 |  | |
| Purpose | Present an information box in medium blue with White background. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: none  Size: MEDIUM  Color: MediumBlue  Location: C |  |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1) Backdrop White  (defined for the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box is presented to the player; the information box fades in and then fades out.**  Start Screen 2  Play Screen 2  FADE IN Information Box 1 as a QUICK EFFECT  FADE OUT Information Box 1 as a QUICK EFFECT  End Screen 2 | |  |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

**Table 5 Act 1, Scene 1, Screen 3**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 3 |  | |
| Purpose | Present a dark blue information box with white background. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: none  Size: MEDIUM  Color: DarkBlue  Location: C |  |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1) Backdrop White  (defined for the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box is presented to the player; the information box fades in and then fades out.**  Start Screen 3  Play Screen 3  FADE IN Information Box 1 as a QUICK EFFECT  FADE OUT Information Box 1 as a QUICK EFFECT  End Screen 3  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | |  |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

## Test Game 3 Requirements

**Table 1 Act 1**

|  |  |
| --- | --- |
| Identifier | Act 1. |
| Purpose | Start, Play, and End Act 1. |
| Learning objectives | Not present. |
| Uses Scenes | Scene 1  Scene 2  Straight Cut into, out of Scene 1 and Scene 2. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Act |

**Table 2 Act 1, Scene 1**

|  |  |
| --- | --- |
| Identifier | Scene 1. |
| Purpose | Start, Play, and End Screen 1, Screen 2 with a BlueSky backdrop. |
| Learning objectives | Not present. |
| Uses Screens | Screen 1  Screen 2  Straight Cut into, out of Screen 1 and Screen 2. |
| Backdrop | Bluesky. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for Scene 1  START SCENE 1  PLAY SCENE 1  END SCENE 1  Final state for Scene 1 |
| Alternate flow of events | If an error occurs, then end the game. |

**Table 3 Act 1, Scene 1, Screen 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 1. |  | |
| Purpose | Present an information box and a button to the player. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: Welcome! Test Game 3  Size: MEDIUM  Location: UC | Button 1  Text: Continue…  Size: MEDIUM  Location: DLC |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1. BlueSky backdrop   (defined in the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box and a button are presented.**  Start Screen 1  Play Screen 1  FADE IN the Information Box 1 as a QUICK EFFECT  FADE IN the Button 1 as a QUICK EFFECT  Player clicks Button 1 to continue to the next screen  FADE OUT the Button 1 as a QUICK EFFECT  FADE OUT the Information Box 1 as a QUICK EFFECT  End Screen 1  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | | Continue…  Welcome!  Test Game 3 |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

**Table 4 Act 1, Scene 1, Screen 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 2. |  | |
| Purpose | Present an information box and a button to the player. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: A Softicorp Game  Size: MEDIUM  Location: UC | Button 1  Text: Continue to Scene 2  Size: MEDIUM  Location: DLC |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1. BlueSky Backdrop   (defined in the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box and a button are presented.**  Start Screen 2  Play Screen 2  FADE IN the Information Bubble 1 as a QUICK EFFECT  FADE IN the Button 1 as a QUICK EFFECT  Player clicks Button 1 to continue to the next scene  FADE OUT the Button 1 as a QUICK EFFECT  FADE OUT the Information Bubble 1 as a QUICK EFFECT  End Screen 2  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | | Continue to Scene 2  A SoftiCorp Game |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

**Table 5.2 Act 1, Scene 2**

|  |  |
| --- | --- |
| Identifier | Scene 2. |
| Purpose | Start, Play, and End Screen 1, Screen 2, Screen 3 with a CollegeBuilding backdrop. |
| Learning objectives | Not present. |
| Uses Screens | Screen 1  Screen 2  Screen 3  Straight Cut into, out of Screen 1, Screen 2, and Screen 3. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for Scene 2  START SCENE 2  PLAY SCENE 2  END SCENE 2  Final state for Scene 2 |
| Alternate flow of events | If an error occurs, then end the game. |

**Table 6 Act 1, Scene 2, Screen 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 1 |  | |
| Purpose | Present a CollegeBuilding backdrop |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Not present. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Backdrop | CollegeBuilding |  | |
| Props | | | |
| Generic interaction | Not present. | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1. CollegeBuilding background   (defined in the scene) | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |
| Interactions  (normal flow of events) | Notes. Only the backdrop is presented to the player in this screen.  Start Screen 1  Play Screen 1  The screen is presented for a MODERATE amount of time.  End Screen 1 | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |

**Table 7 Act 1, Scene 2, Screen 2**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 2. |  | |
| Purpose | Present a CollegeBuilding backdrop with a StandClose character. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | NPC: Dr. Ima Coder. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: <character profile>  Color: White  Size: LARGE  Location: UC |  |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1) CollegeBuilding background  (defined in the scene) | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |
| Interactions  (normal flow of events) | **Note. When a character is selected, an information box containing the character’s profile is presented to the player.**  Start Screen 2  Play Screen 2  FADE IN NPC 1 as a QUICK EFFECT  At any time, the Player selects/deselects the Dr. Ima Coder  character to display/un-display the profile.    FADE OUT NPC 1 as a QUICK EFFECT  End Screen 2  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | |  |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |

**Table 8 Act 1, Scene 2, Screen 3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Screen 3. |  | | |
| Purpose | Present a CollegeBuilding backdrop with a StandClose character. |  | | |
| Learning Objectives | Not present. |  | | |
| Declarations, initialization | | | | |
| Challenge | Not present. |  | | |
| Characters | | | | |
| Player | Default Player. |  | | |
| Non-player characters | Not present. |  | | |
| Setting (visual) | | | | |
| Props | | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Conversation Bubble 1  Text: Hello, my name is Dr. Ima Coder, I teach here at Steven B. Allmer School of Software Engineering. Over the course of this game, you will practice a variety of software engineering skills that are commonly used in industry  Size: MEDIUM  Location: UC |  |  | | | | |
| Set Decorations | Not present. | | | |
| Audio | | | | |
| Music | Not present. | |  | |
| Challenge | Not present. | |  | |
| Game Play | | | | |
| Start of Screen | 1) CollegeBuilding background | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png |  |
| Interactions  (normal flow of events) | **Note. A NPC and a conversation bubble are presented to the player.**  Start Screen 2  Play Screen 2  FADE IN NPC 1 as a QUICK EFFECT  FADE IN Conversation Bubble 1 as a QUICK EFFECT  Present Conversation Bubble 1 for a MODERATE amount of  time  FADE OUT the Conversation Bubble 1 as a QUICK EFFECT  FADE OUT NPC 1 as a QUICK EFFECT  End Screen 3  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | |  | |
| Alternate flow of events | If an error occurs, then end the game. | | | |
| End of Screen |  | | C:\Users\Cooper\Dropbox\ SimSYS\GameAssetRepository\Visual\Backdrops\CollegeBuilding.png | |

**Table 9 Act 2**

|  |  |
| --- | --- |
| Identifier | Act 2. |
| Purpose | Start, Play, and End Act 1. |
| Learning objectives | Not present. |
| Uses Scenes | Scene 1  Straight Cut into, out of Scene 1. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Game |

**Table 10.1 Act 2 Scene 1**

|  |  |
| --- | --- |
| Identifier | Scene 1. |
| Purpose | Start, Play, and End Screen 1, Screen 2 with a BlueSky backdrop. |
| Learning objectives | Not present. |
| Uses Screens | Screen 1  Straight Cut into, out of Screen 1. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for Scene 1  START SCENE 1  PLAY SCENE 1  END SCENE 1  Final state for Scene 1 |

**Table 11 Act 2 Scene 1 Screen 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 1. |  | |
| Purpose | Present a BlueSky backdrop. |  | |
| Learning Objectives | Not present. |  | |
| Declarations, initialization | | | |
| Challenge | Not present. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Backdrop | BlueSky |  | |
| Props | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: Thank you for playing Test Game 3!  Size: MEDIUM  Location: UC | Button 1  Text: End  Size: MEDIUM  Location: DLC |  | | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Challenge | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1. BlueSky backdrop   (defined in the scene) | |  |
| Interactions  (normal flow of events) | **Note. An information box and a button are presented to the player.**  Start Screen 1  Play Screen 1  FADE IN Information Box 1 as a QUICK EFFECT  FADE IN Button 1 as a QUICK EFFECT  Player clicks Button 1 to end the game.  FADE OUT the Information Box 1 as a QUICK EFFECT  FADE OUT the Button 1 as a QUICK EFFECT  End Screen 1  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | | End…  Thank you for playing  Test Game 3! |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

## Test Game 4 Requirements

**Table 1 Act 1**

|  |  |
| --- | --- |
| Identifier | Act 1 |
| Purpose | Start, Play, and End Act 1. |
| Learning objectives | Not present. |
| Uses Scenes | Scene 1.  Straight Cut into, out of Scene 1. |
| Behaviour  state machine description:  Current  Transition  event  condition  output  Next | Initial state for the Act  START ACT 1  PLAY ACT 1  END ACT 1  Final state for the Game |
| Alternate flow of events | If an error occurs, then end the game. |

**Table 2 Act 1, Scene 1**

|  |  |
| --- | --- |
| Identifier | Act 1 Scene 1 |
| Purpose | Start, Play, and End Screen 1 and Screen 2. |
| Learning objectives | Not present. |
| Uses Screens | Screen1  Screen 2  Straight Cut into, out of Screen 1 and Screen 2. |
| Backdrop | BlueSky |
| Behaviour  state machine description:  Current screen  Transition  event  condition  output  Next screen | Initial state for Scene 1  START SCENE 1  PLAY SCENE 1  END SCENE 1  Final state for Scene 1 |
| Alternate flow of events | If an error occurs, then end the game. |

**Table 3 Act 1, Scene 1, Screen 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Screen 1 |  | |
| Purpose | Present a challenge (multiple choice quiz) to the player, which awards points. |  | |
| Learning Objectives | Software engineering  SWEBOK = Software Design  General Design Concepts  Context of Software Design  Software Design Process  Bloom = Knowledge, Comprehension |  | |
| Declarations, initialization | | | |
| Uses Challenges | Challenge 1.  Straight Cut into, out of Challenge 1. |  | |
| Characters | | | |
| Player | Default Player. |  | |
| Non-player characters | Not present. |  | |
| Setting (visual) | | | |
| Props | | | |
| Generic interaction | Not present. | | |
| Set Decorations | Not present. | | |
| Audio | | | |
| Music | Not present. | |  |
| Game Play | | | |
| Start of Screen | 1) Backdrop BlueSky  (defined for the scene) | |  |
| Interactions  (normal flow of events) | **Note. A quiz is presented to the player.**  Start Screen 1  Play Screen 1  Challenge 1  End Screen 1  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | | See Challenge 1 |
| Alternate flow of events | If an error occurs, then end the game. | | |
| End of Screen |  | |  |

**Table 4. Act 1, Scene 1, Screen 1, Challenge 1 (quiz)**

|  |  |  |
| --- | --- | --- |
| Identifier | Challenge 1 | |
| Purpose | Multiple choice quiz on design patterns. | |
| Learning Objectives | Software Design  Challenge = Interactive Quiz  SWEBOK = Software Design  General Design Concepts  Context of Software Design  Software Design Process  Bloom = Knowledge, Comprehension  Understand the purpose of the bridge design pattern. | |
| Character presenting the quiz | None. | |
| Competing Characters | None. | |
| Quiz Style | Layout (overall organization)  Multiple choice  Question (stem description, stem question) on the left  Answer options on the right (two or more) | Generic interaction props  Question: information bubbles  Answer: button  Hint: information box  Feedback: information bubble |
| Reward scheme | Correct answer: add 500 points.  Incorrect answer: no change to points. | |
| Declarations, initialization | | |
| Quiz interaction props | |  |  |  | | --- | --- | --- | | ***Stem***  Stem description Information Bubble 1  Text: The designers need to use a complex AI algorithm that is available as a 3rd party component with an API. The designers know there are a number of possible components available on the marketplace, which are undergoing extensive performance evaluations. The final recommendation will not be available for several months.  Stem Question Information Bubble 2  Text: Which design pattern would be suitable to reduce the impact of this eventual recommendation?  ***Player Feedback***  Feedback Information Bubble 3  Text: The bridge pattern is correct!  Feedback Information Bubble 4  Text: Sorry – the correct answer is the bridge pattern  ***Hints***  Hint Information Box 1  Text: The Strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable; algorithms are selected at runtime. It is a behavioral pattern.  Hint Information Box 2  Text: The Bridge pattern decouples an interface and its implementation. The implementation can be modified without changing the interface. It is a structural pattern.  Hint Information Box 3  Text: The Factory pattern defines an interface for creating an object, but lets the classes that implement the interface decide which class to instantiate. It is a creational pattern.  Hint Information Box 4  Text: The Singleton pattern that restricts the instantiation of a class to only one object. It is a creational pattern. | ***Options***  Answer Option Button 1  Text: Strategy pattern  Hint: Hint Information Box 1  Evaluation: incorrect  Player Feedback: Feedback Information Bubble 4  Answer Option Button 2  Text: Bridge pattern  Hint: Hint Information Box 2  Evaluation: correct  Player Feedback: Feedback Information Bubble 3  Answer Option Button 3  Text: Factory pattern  Hint Information Box 3  Evaluation: incorrect  Player Feedback: Feedback Information Bubble 4  Answer Option Button 4  Text: Singleton pattern  Hint Information Box 4  Evaluation: incorrect  Player Feedback: Feedback Information Bubble 4 |  | | |
| Timers | None. | |
| Game Play | | |
|  | | |

|  |  |  |
| --- | --- | --- |
| Interactions  (normal flow of events) | **Note. Challenge 1 (a multiple choice quiz) is presented.**  Start Challenge 1  Play Challenge 1  ***Present question and answers***  FADE IN stem description, stem question, and answers as  a QUICK EFFECT  ***Player Displays Hint***  If the player moves the mouse over an answer option, display the hint  If the player moves the mouse away from an answer, remove the hint  from the display  ***Players Answers***  FADE OUT stem description, stem question, and answers as  a QUICK EFFECT  ***Evaluate Answers***  Answer option 1: no change  Answer option 2: add 500 points for the player  Answer option 3: no change  Answer option 4: no change  ***Present player feedback***  If answer option 1, then display the feedback for a MODERATE  amount of time.    If answer option 2, then display the feedback for a MODERATE  amount of time.    If answer option 3, then display the feedback for a MODERATE  amount of time.    If answer option 4, then display the feedback for a MODERATE  amount of time.    End Challenge 1  **Note. At the end of the challenge, the game needs to remove the visual setting from the display.** | Option  Option  Option  Option  Stem question  Stem description  Feedback |

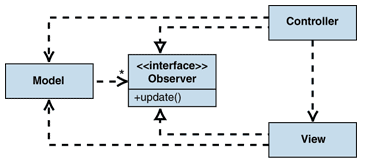
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Screen 2 | | |  |
| Purpose | Present feedback about their title and points; present a button to end the game. | | |  |
| Learning Objectives | Not present. | | |  |
| Declarations, initialization | | | | |
| Challenge | Not present. |  | | |
| Characters | | | | |
| Player |  |  | | |
| Non-player characters |  |  | | |
| Setting (visual) | | | | |
| Props | | | | |
| Generic interaction | |  |  |  | | --- | --- | --- | | Information Box 1  Text: Your title is <title> and you have <points> points!  Size: MEDIUM  Location: UC | Button 1  Text: End…  Size: MEDIUM  Location: DLC |  | | | | |
| Set Decorations | Not present. | | | |
| Audio | | | | |
| Music | Not present. | |  | |
| Challenge | Not present. | |  | |
| Game Play | | | | |
| Start of Screen | 1) Backdrop BlueSky  (defined for the scene) | |  | |
| Interactions  (normal flow of events) | **Note. The game is wrapped up by presenting the title and number of points to the player. A button to end the game is presented.**  Start Screen 2  Play Screen 2  FADE IN the Information Box 1 as a QUICK EFFECT  FADE IN the Button 1 as a QUICK EFFECT  Player clicks Button 1 to end the game.  FADE OUT the Button 1 as a QUICK EFFECT  FADE OUT the Information Box 1 as a QUICK EFFECT  End Screen 2  **Note. At the end of the screen, the game needs to remove the visual setting from the display.** | | Your title is Junior Programmer and you have 1500 points!  End… | |
| Alternate flow of events | If an error occurs, then end the game. | | | |
| End of Screen |  | |  | |

## Game Play Engine Architecture

### Style

The architecture used for the project is Model-View-Controller pattern with passive model. 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:

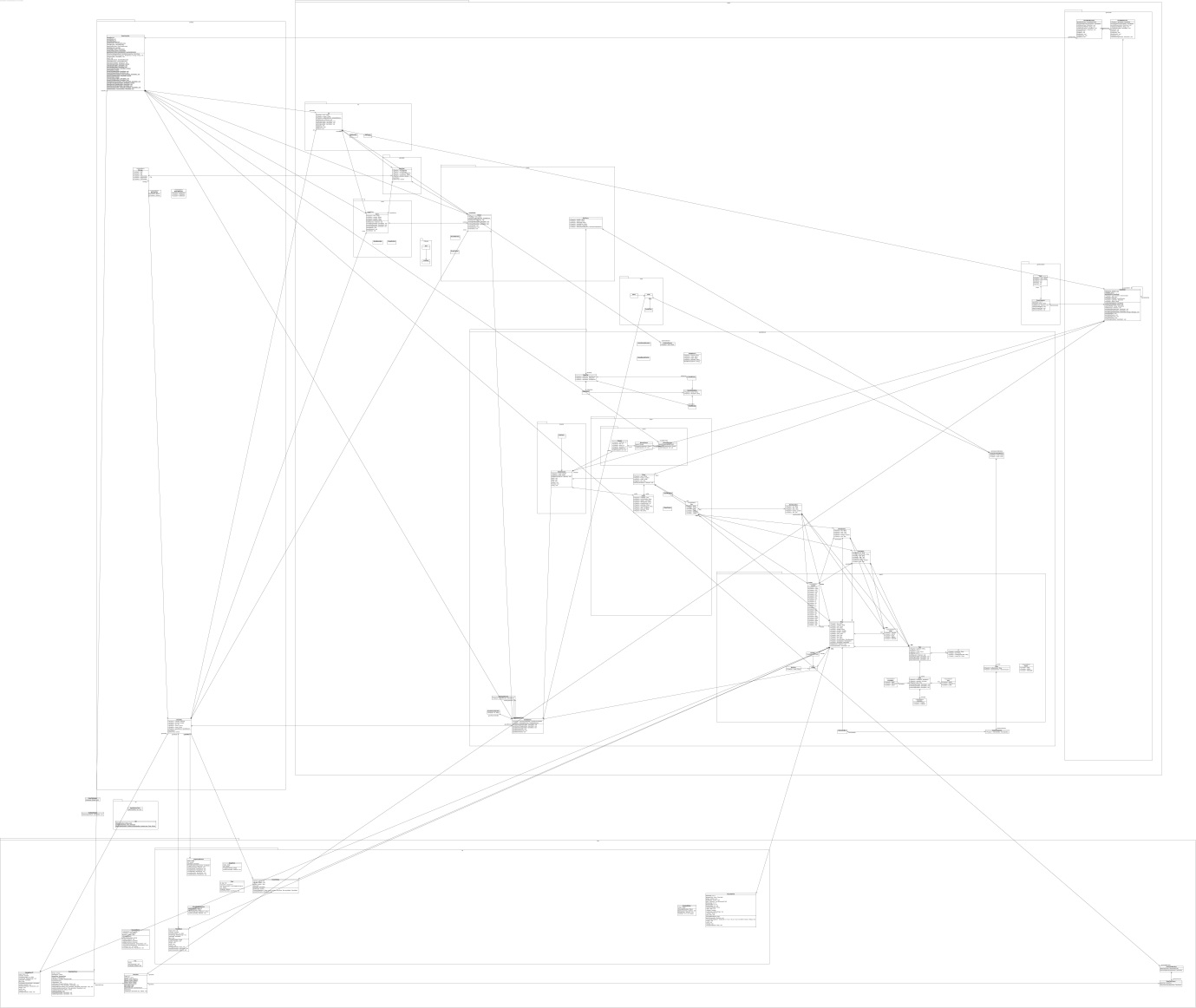
1. Model: The model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller).
2. View: The view manages the display of information.
3. Controller: The controller interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate.



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.

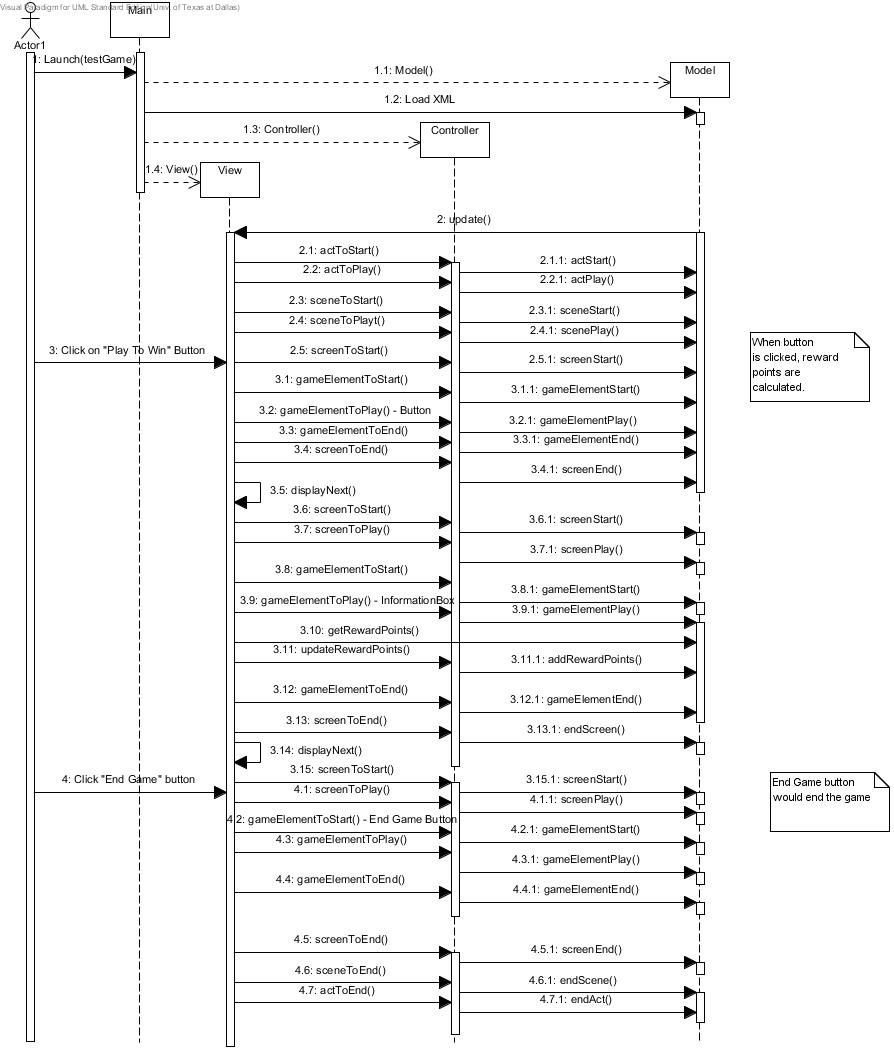
Also the view must notify the model, if the update pushed to the model is complete. This is a place where, the controller comes to play. The logic that is specific to the application is present in the controller and not in the model. By keeping the logic in the controller, the views and models can remain 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.

### Structural View (Component Diagram)

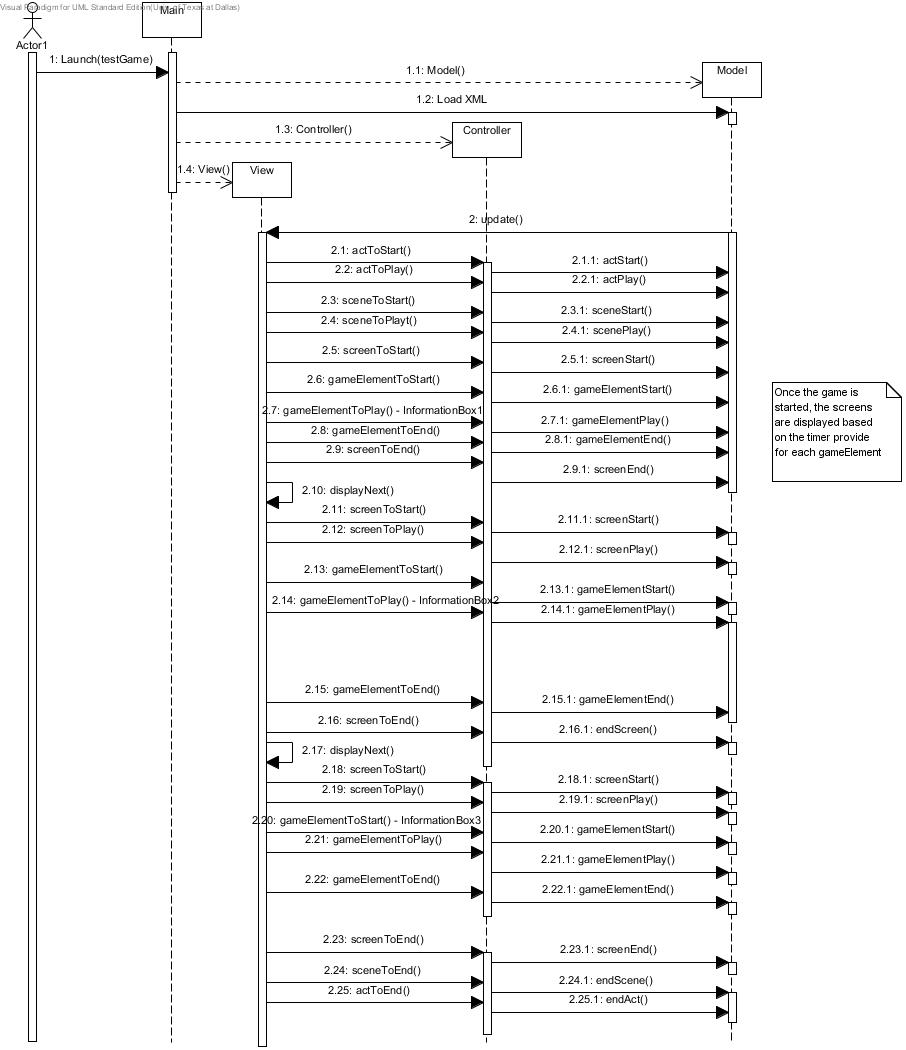


### Behavioural View (Sequence Diagram)

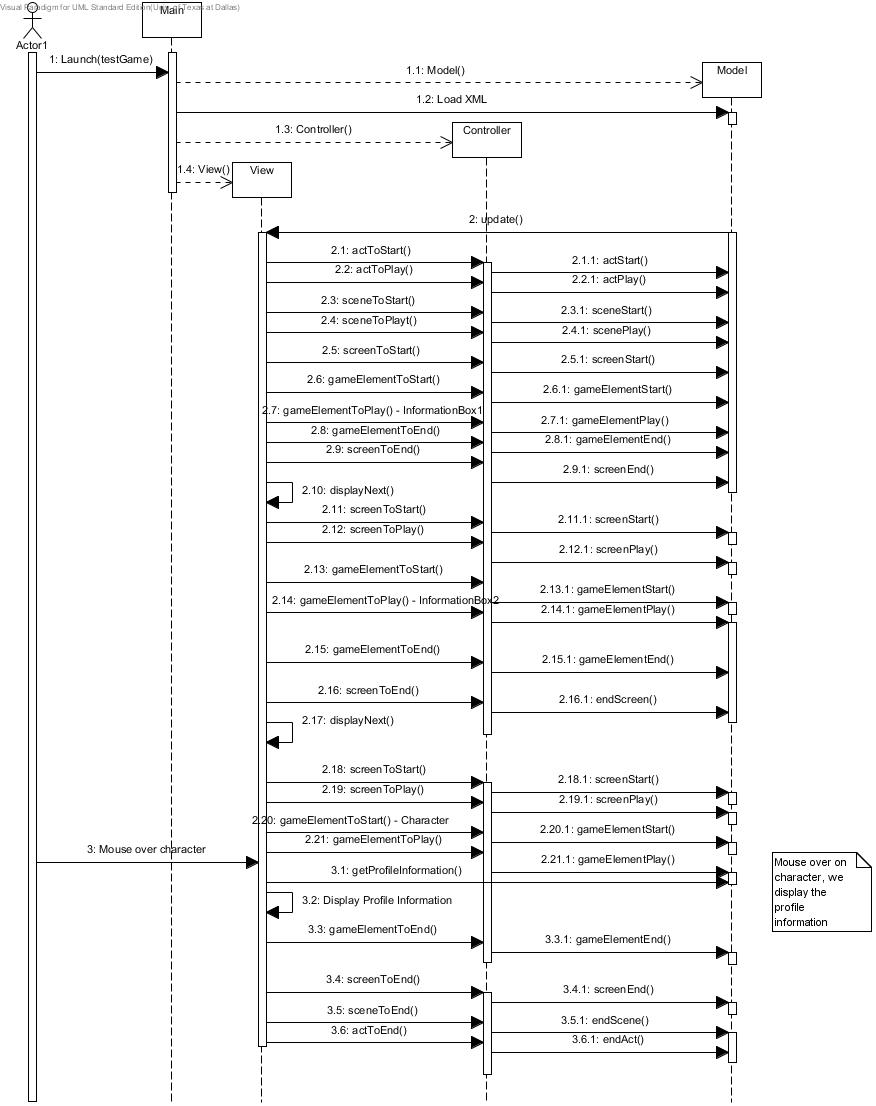
TestGame1



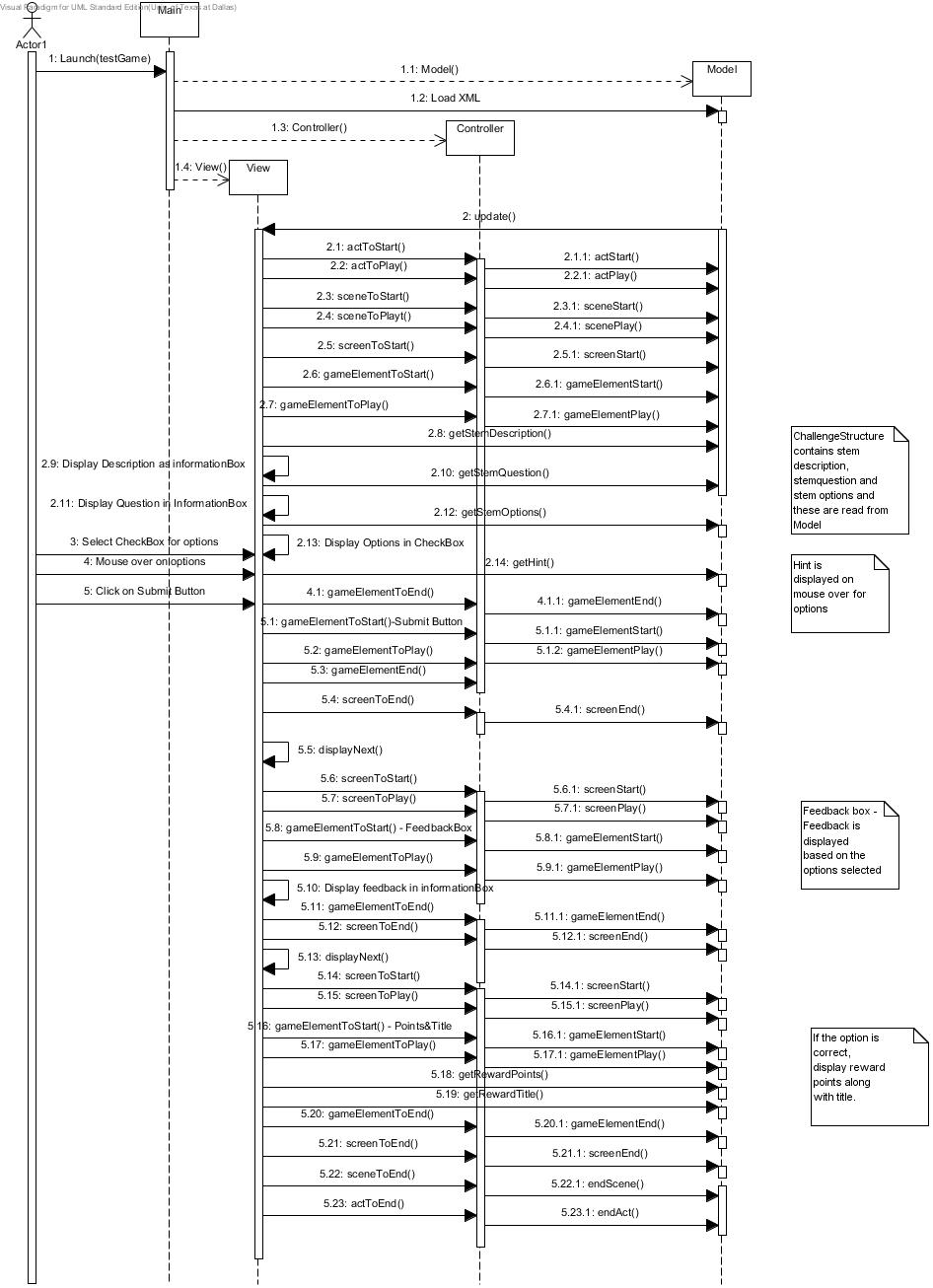
TesstGame2



TestGame3



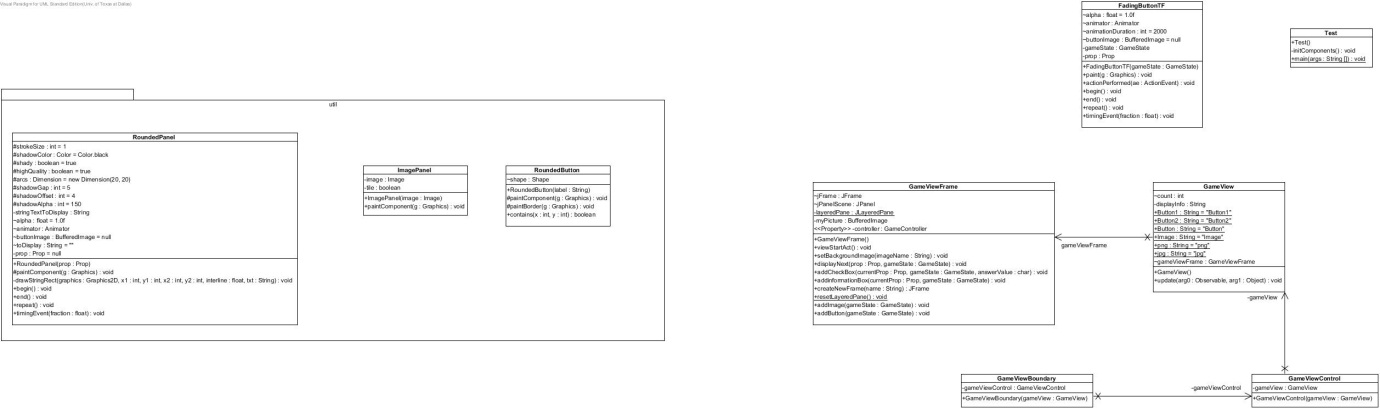
TestGame4



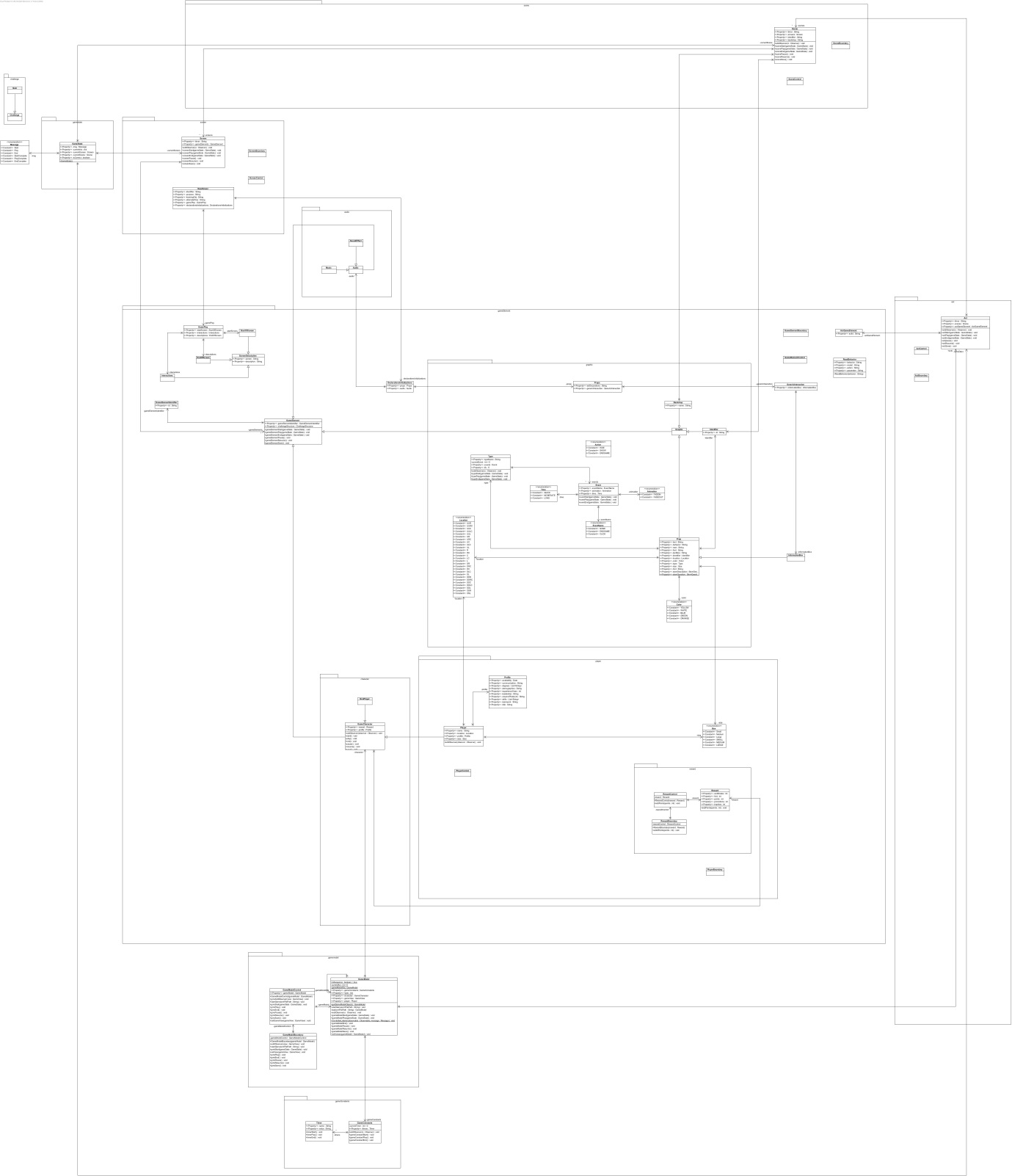
## Game Play Engine Design

### Structural View (Class Diagram)

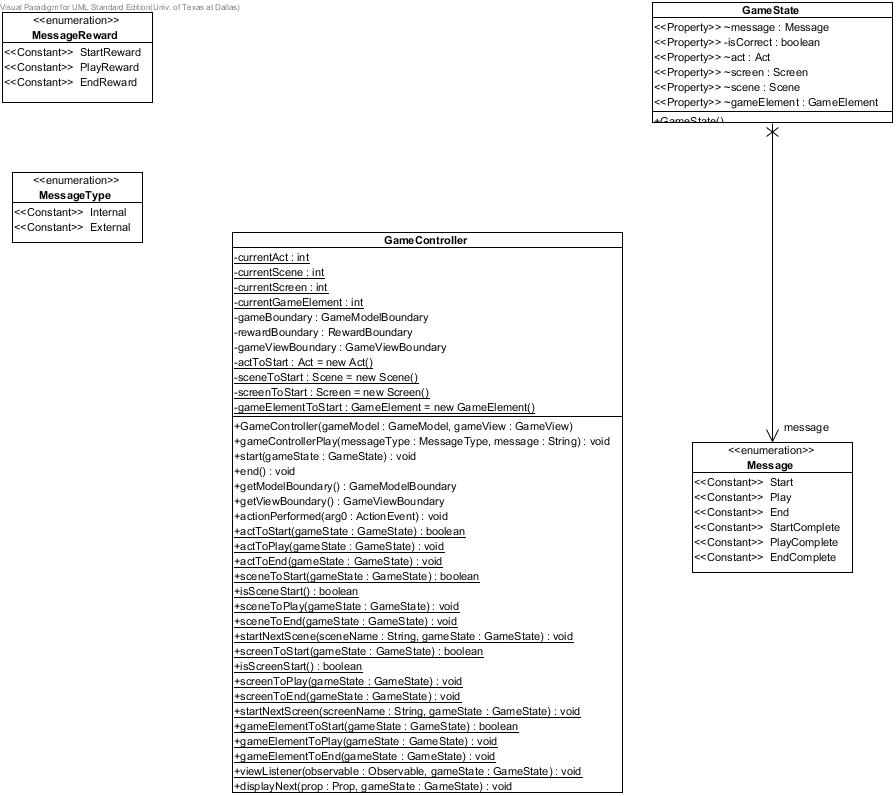
**Game View**

****

**Game Model**

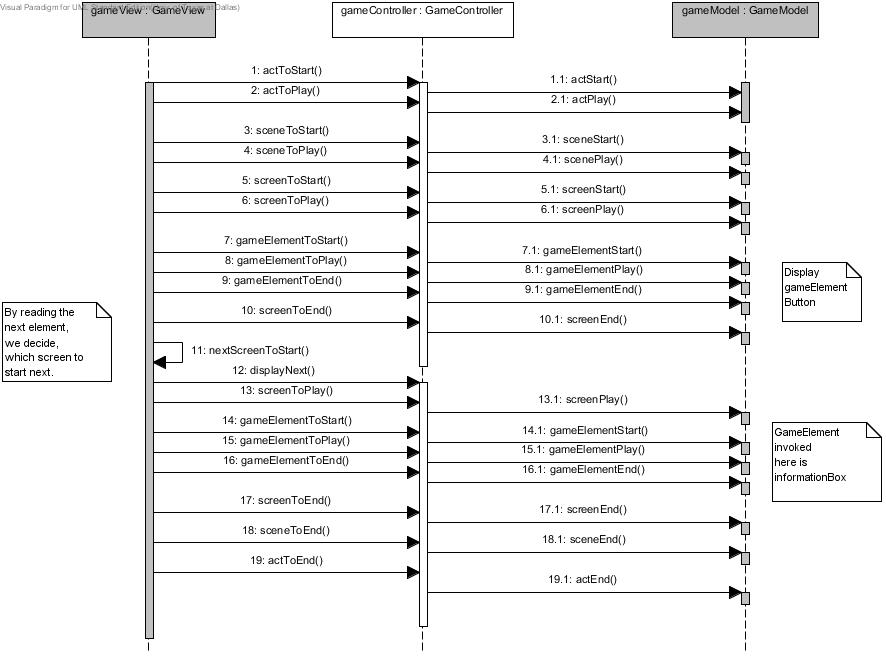


**Game Controller**

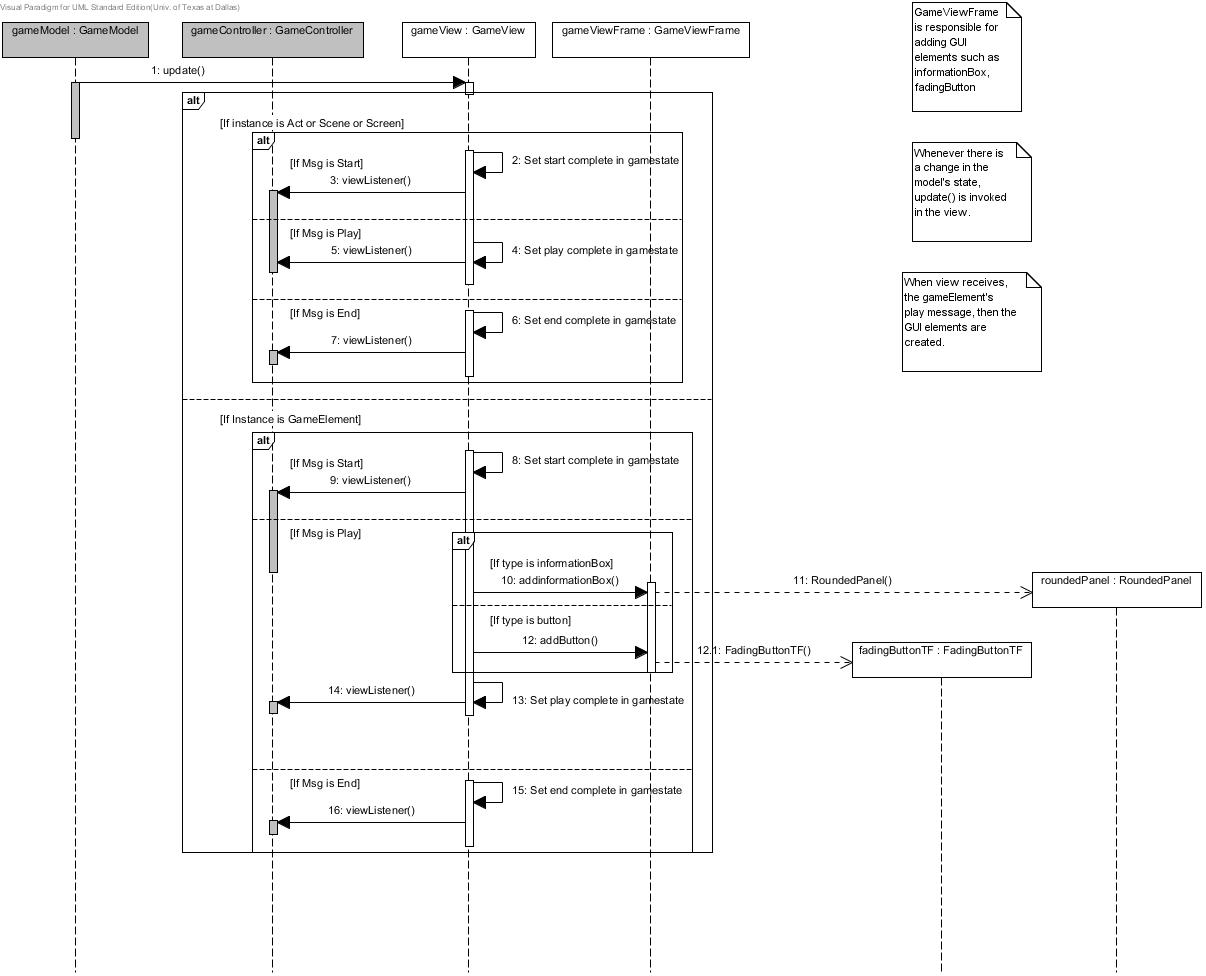


### Behavioural Views (Sequence Diagram)

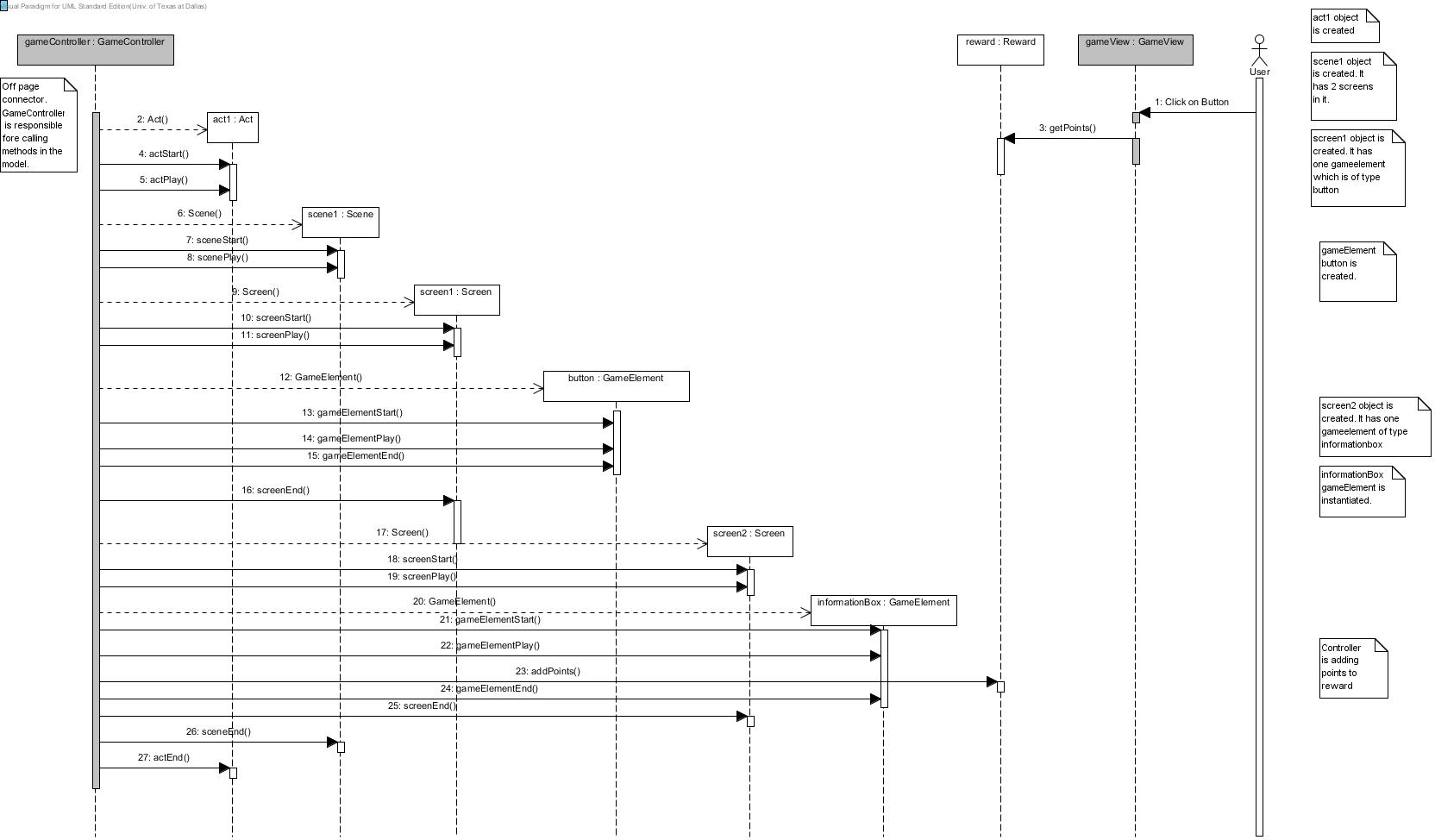
TestGame1- Controller



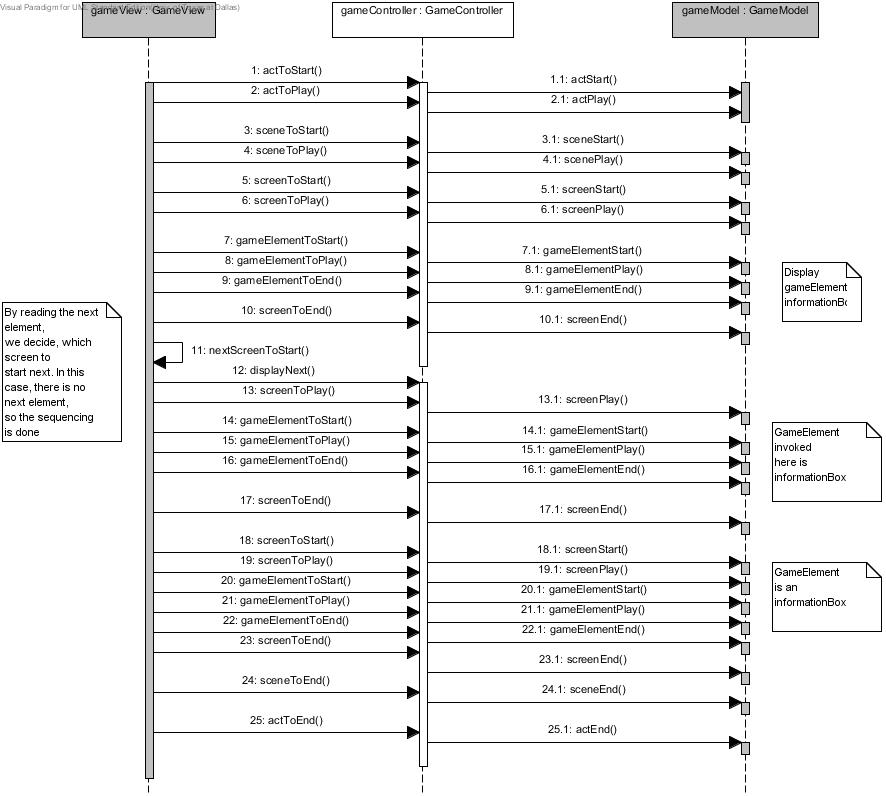
TestGame1-View



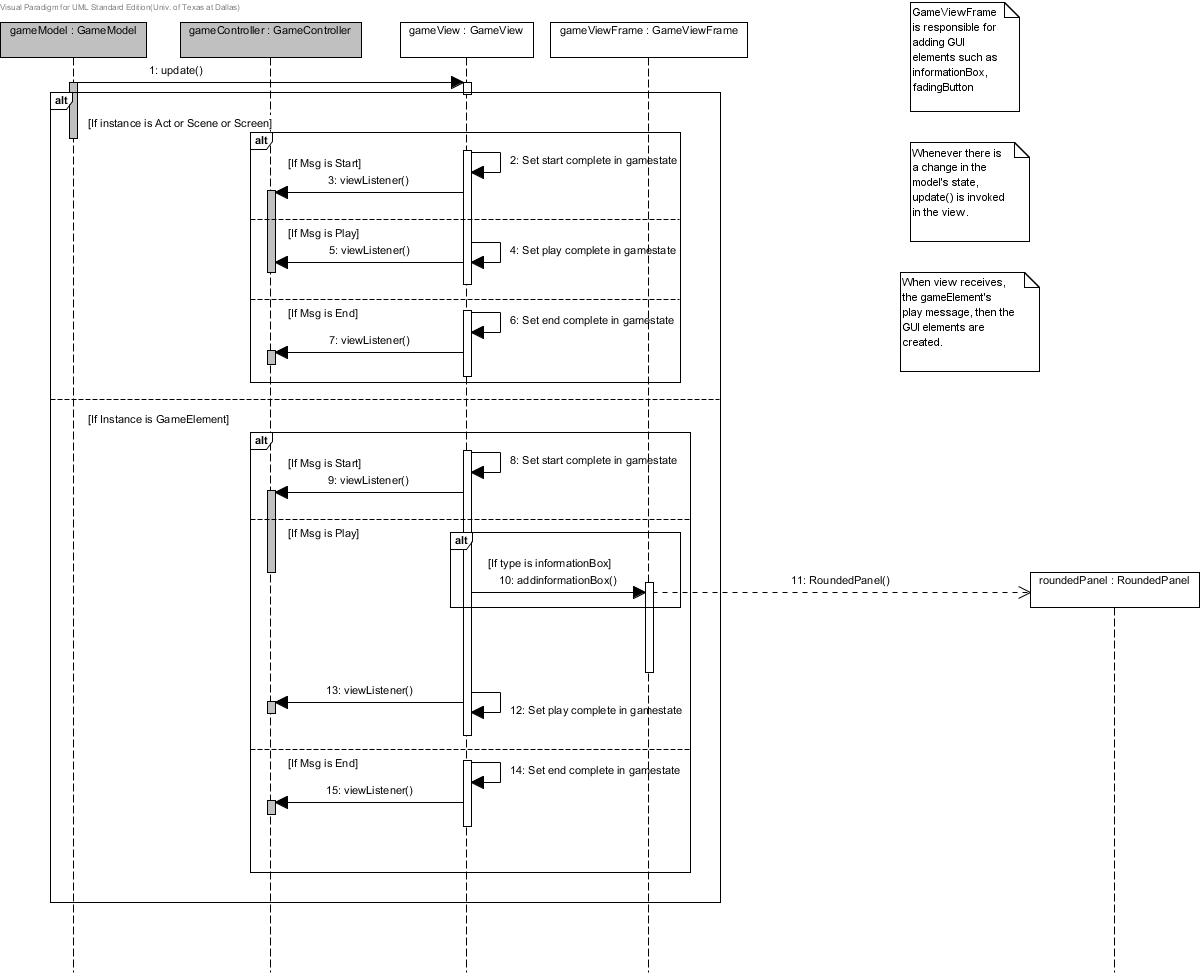
TestGame1-Model



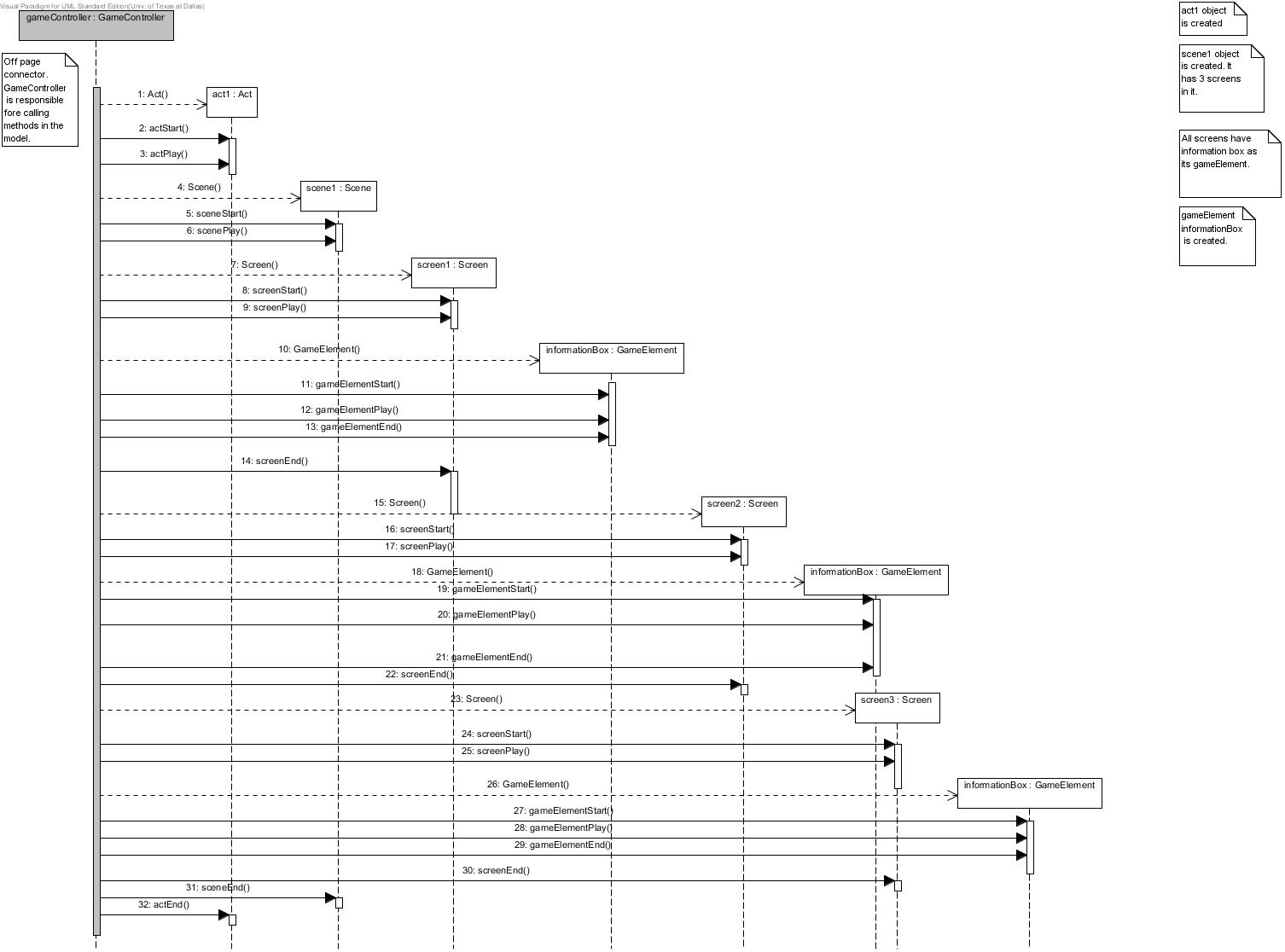
TestGame2- Controller



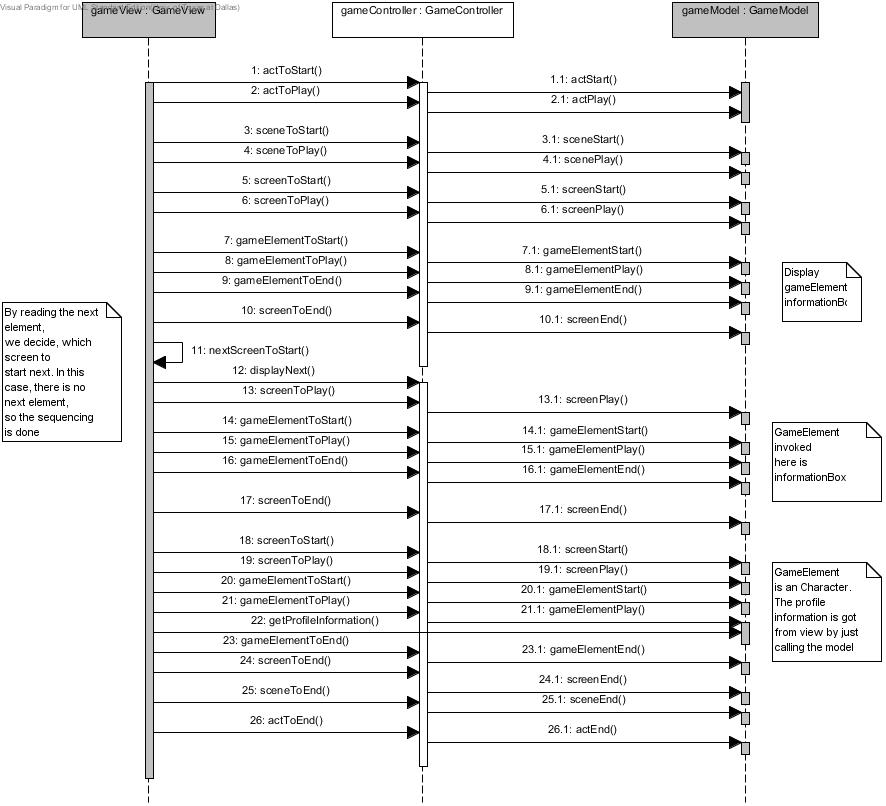
TestGame2-View



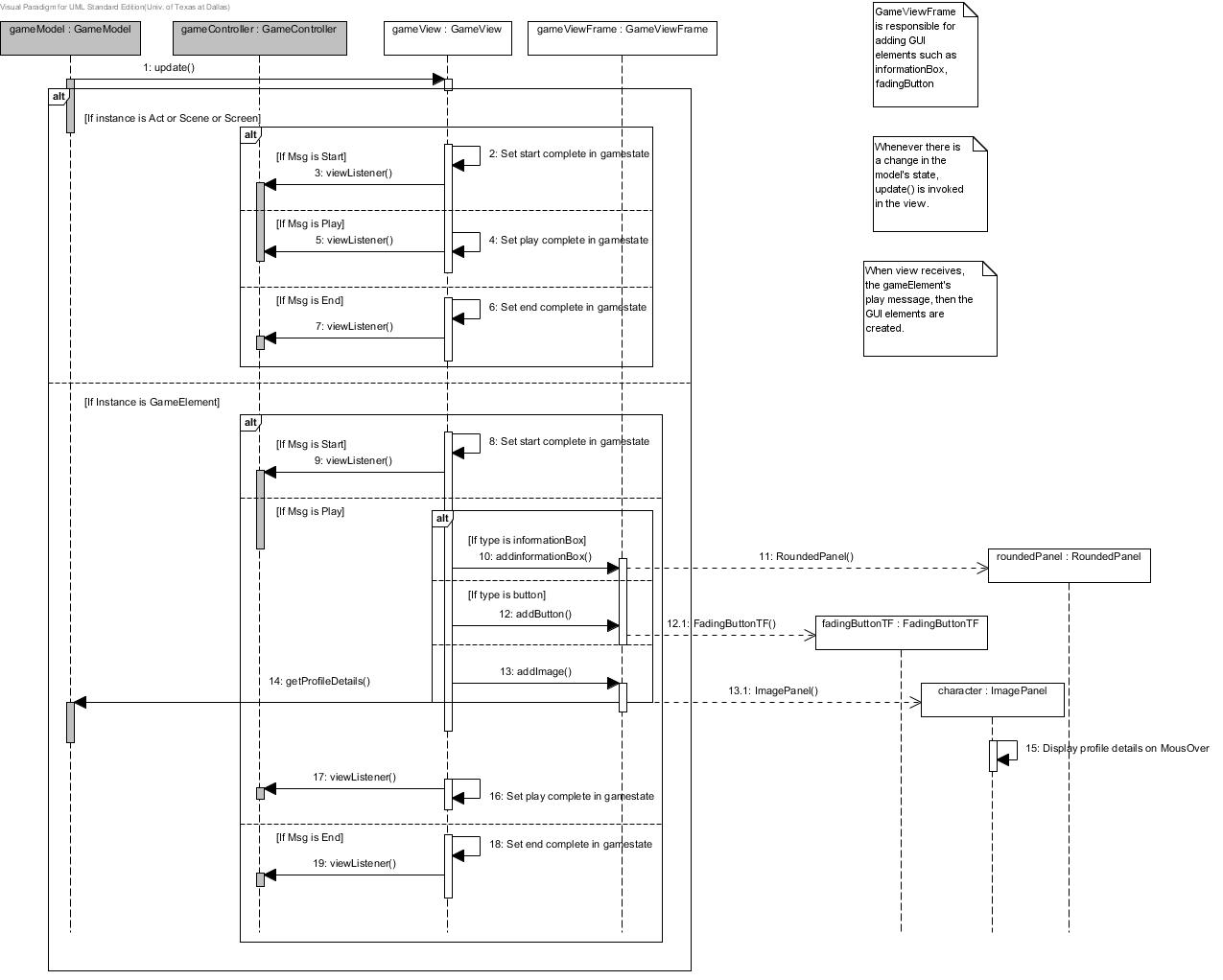
TestGame2-Model



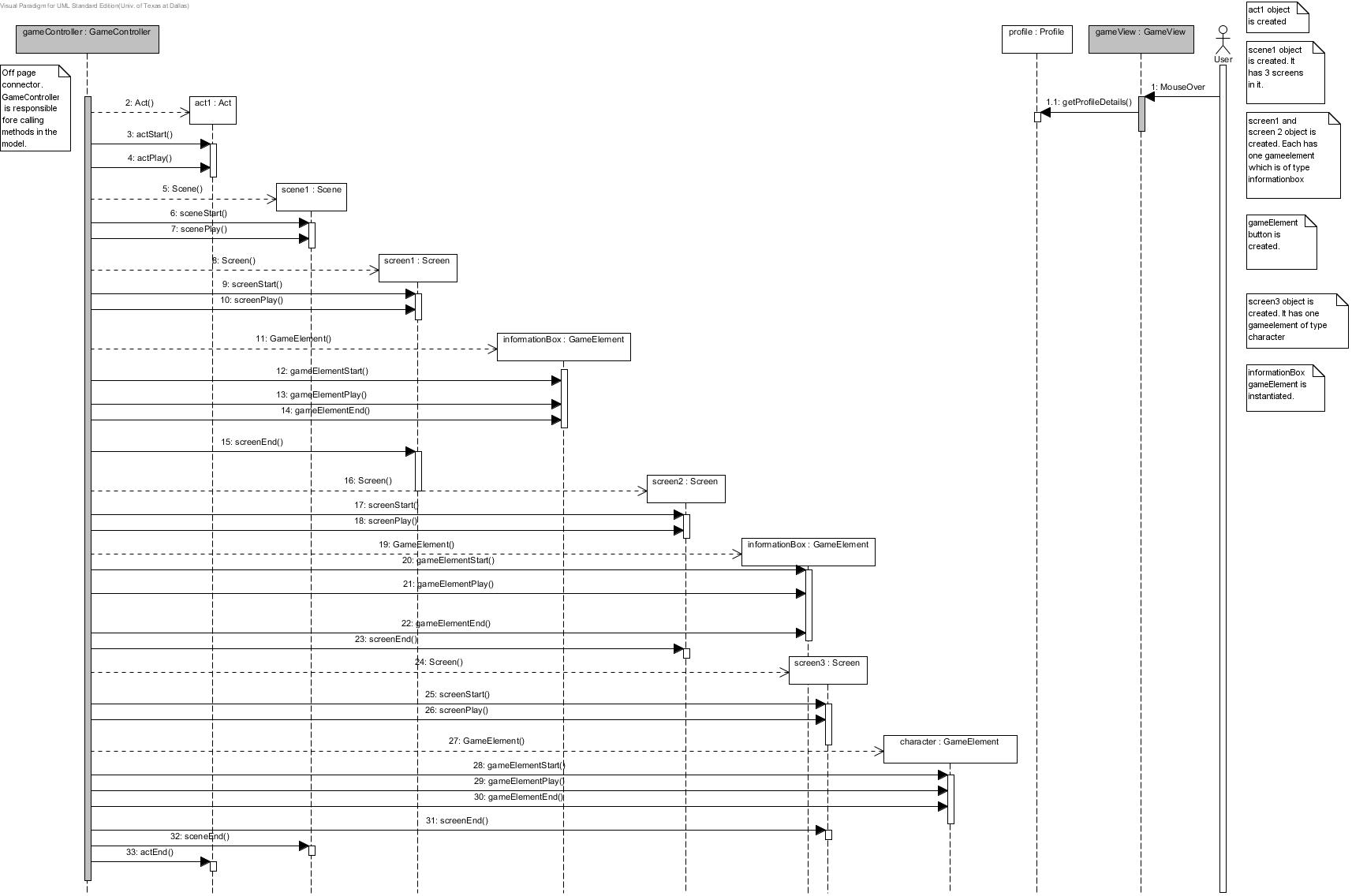
TestGame3- Controller



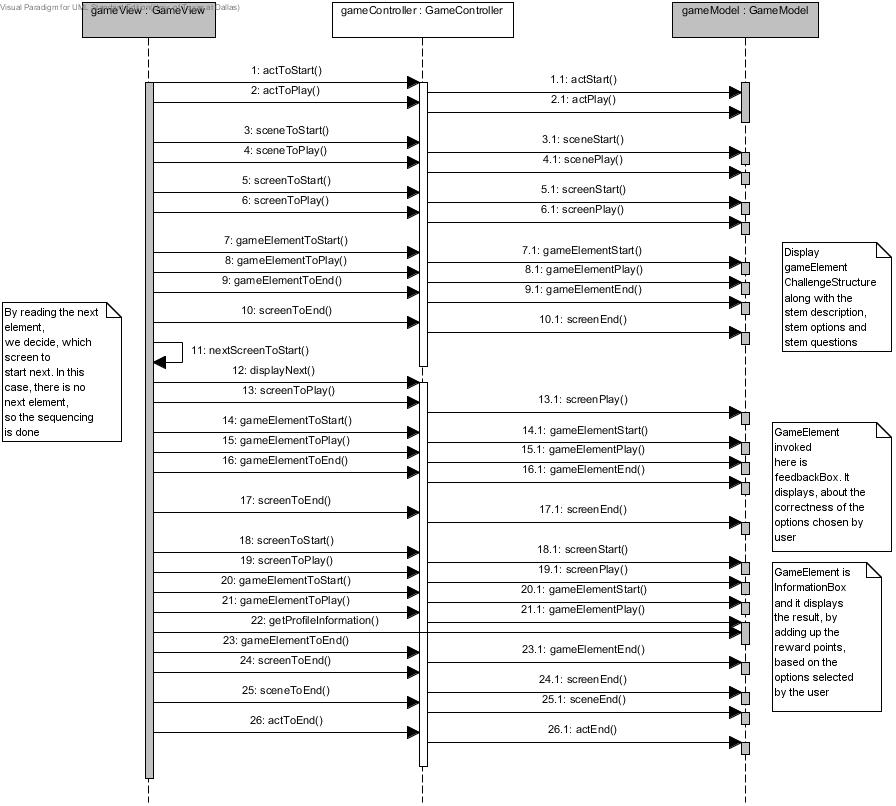
TestGame3-View



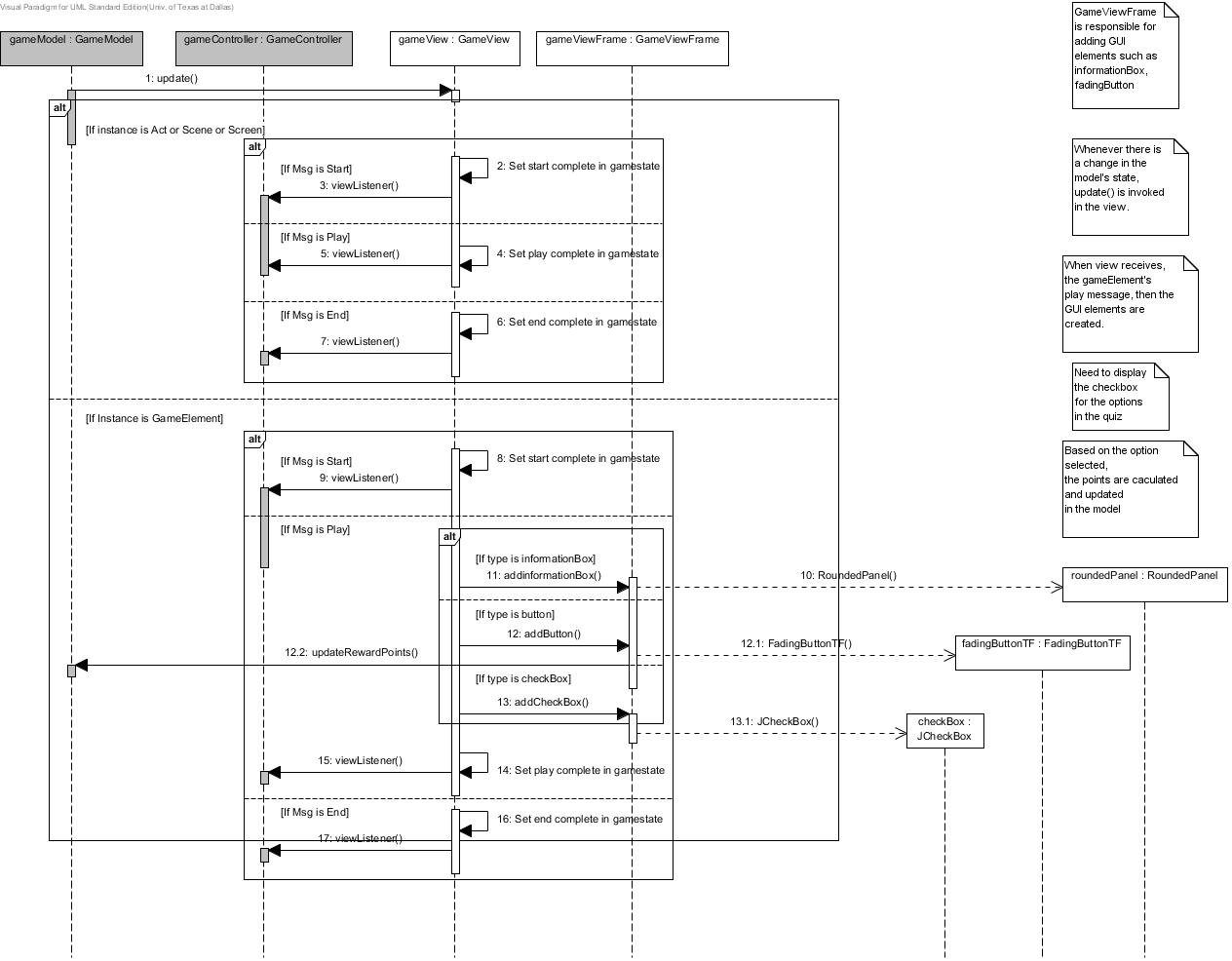
TestGame3-Model



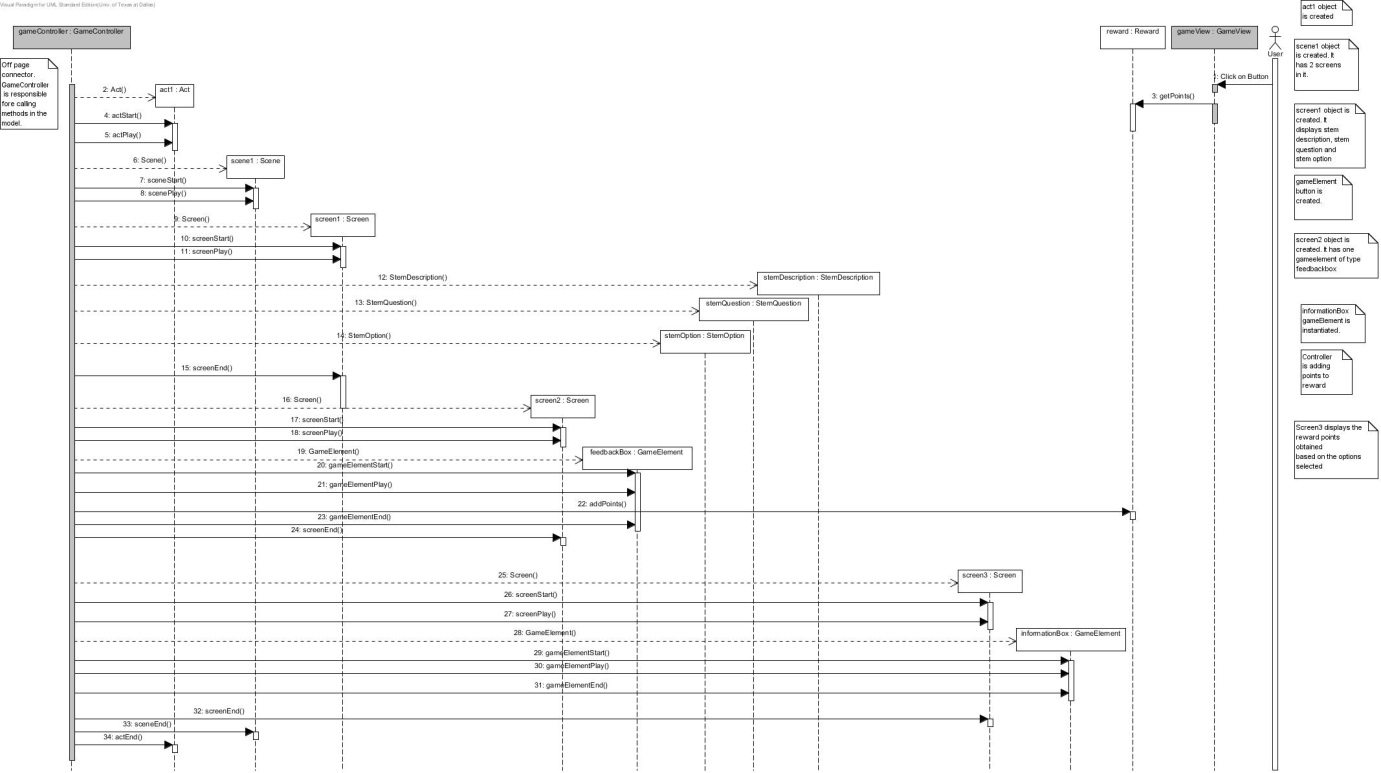
TestGame4- Controller



TestGame4-View



TestGame4-Model



## Code

### Coding Standards, Sample Code, Libraries Used

**Coding Standards**

1. Names representing packages should be in all lower case. (E.g gameelement.model.act)
2. Names representing types must be nouns and written in mixed case starting with upper case. (E.g GameElement)
3. Variable names must be in mixed case starting with lower case. (E.g gameState)
4. Names representing constants (final variables) must be all uppercase using underscore to separate words. (E.g QUICK, MODERATE)
5. Names representing methods must be verbs and written in mixed case starting with lower case. (E.g. getProfile())
6. Generic variables should have the same name as their type. (E.g void getProfile(Profile profile)
7. All names should be written in English.
8. Variables with a large scope should have long names, variables with a small scope can have short names
9. The name of the object is implicit, and should be avoided in a method name. (line.getLength(); // NOT: line.getLineLength();
10. The terms *get/set* must be used where an attribute is accessed directly.(E.g .private String text, //getter is getText(), setter is setText())
11. The term *initialize* can be used where an object or a concept is established.(E.g gameElement.initializeFontSet())
12. Java source files should have the extension *.java*. (GameElement.java)
13. File content must be kept within 80 columns.
14. Imported classes should always be listed explicitly. (E.g import java.util.List; // NOT: import java.util.\*;)
15. Class variables should never be declared public.

**Library Used**

1. **TimingFramework:-**

The Timing Framework is a set of utility classes that provides a much more capable animation system that handles many of the details that you would other-wise need to implement in your application.

The motivations for all of the features in the Timing Framework were twofold:

**• Handle common tasks:** Much of the code that we write in animating graphics and GUIs is necessary for nearly all animations. For example, most time-based animations need to figure out what fraction of the animation has completed at any given time during the animation, so why not simplify things by calculating that fraction automatically.

**• Simple API:** In providing more capabilities for animations, we do not want to create an API that is prohibitively complex. It should be as easy to use as possible. The framework has a few distinct levels of functionality. At the core of the framework is the timing package, with the fundamental building blocks that all of the other pieces use.

An additional level of functionality is provided in the triggers and interpolation packages. Triggers associate animations with specific events and automate starting animations on the basis of those events. Property setters in the interpolation package provide the ability to animate properties of Java objects and to define complex models of how those properties are interpolated between different values.

1. **JAXB:-**

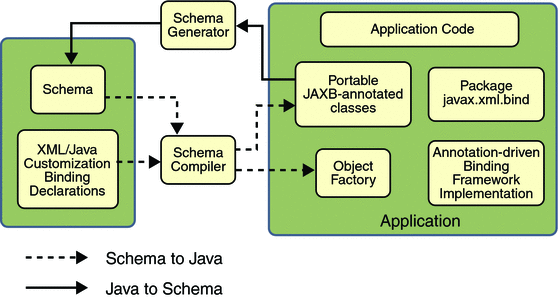
JAXB is an acronym derived from [*Java Architecture for XML Binding.*](http://java.sun.com/developer/technicalArticles/WebServices/jaxb) It constitutes a convenient framework for processing XML documents, providing significant benefits as compared to previously available methods such as the one following the [Document Object Model (DOM)](http://www.w3.org/DOM/). In the DOM approach, the parser creates a tree of objects that represents the content and organization of data in the document. The application can then navigate through the tree in memory to access the data it needs. DOM data, however, is contained in objects of a single type, linked according to the XML document's structure, with individual node objects containing an element, an attribute, a CDATA section, etc. Values are invariably provided as strings.

Unmarshalling an XML document with the appropriate JAXB method also results in a tree of objects, with the significant difference being that the nodes in this tree correspond to XML elements, which contain attributes and the content as instance variables and refer to child elements by object references.

JAXB uses Java's annotations for augmenting the generated classes with additional information that bridges the gap between what is described by an XML schema and the information available (via Java's reflection mechanisms) from a set of Java class definitions. Adding such annotations to existing Java classes prepares them for being used by JAXB's runtime.

A JAXB implementation consists of the following architectural components:

* Schema compiler: Binds a source schema to a set of schema-derived program elements. The binding is described by an XML-based binding language.
* Schema generator: Maps a set of existing program elements to a derived schema. The mapping is described by program annotations.
* Binding runtime framework: Provides unmarshalling (reading) and marshalling (writing) operations for accessing, manipulating, and validating XML content using either schema-derived or existing program elements.



1. **Java Swing:-**

Swing is the primary [Java](http://en.wikipedia.org/wiki/Java_(programming_language)) [GUI](http://en.wikipedia.org/wiki/Graphical_user_interface) [widget toolkit](http://en.wikipedia.org/wiki/Widget_toolkit). It is part of [Oracle](http://en.wikipedia.org/wiki/Oracle_Corporation)'s [Java Foundation Classes](http://en.wikipedia.org/wiki/Java_Foundation_Classes) (JFC) — an [API](http://en.wikipedia.org/wiki/Application_programming_interface) for providing a [graphical user interface](http://en.wikipedia.org/wiki/Graphical_user_interface) (GUI) for Java programs. Swing was developed to provide a more sophisticated set of GUI [components](http://en.wikipedia.org/wiki/Software_component) than the earlier [Abstract Window Toolkit (AWT)](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit). Swing provides a native [look and feel](http://en.wikipedia.org/wiki/Look_and_feel) that emulates the look and feel of several platforms, and also supports a [pluggable look and feel](http://en.wikipedia.org/wiki/Pluggable_look_and_feel) that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

Swing is a platform-independent, [Model-View-Controller](http://en.wikipedia.org/wiki/Model-View-Controller) [GUI](http://en.wikipedia.org/wiki/GUI) framework for Java, which follows a single-[threaded](http://en.wikipedia.org/wiki/Thread_(computing)) programming model. Additionally, this framework provides a layer of abstraction between the code structure and graphic presentation of a Swing-based GUI.

## Testing

### Testing Approach

In this project, the testing was performed by creating variations of test game XML. Ten variations are created for each test game and the code is tested against these variations. Also, after testing a current game, regression testing was performed on all the previous test games. The main objective of regression testing was not to break any code that was previously written.

### Test Game 4 Sample Test Case

**Screen 1**

**TestGame1- screenshot**

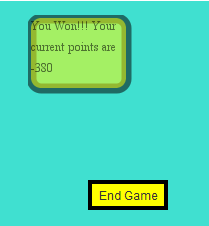
**Screen-1 background**

****

**Screen-1 GameElement**

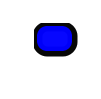
****

**Screen-2 (2 GameElements)**

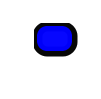
****

**TestGame2 – screenshots**

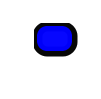
**Screen-1**

****

**Screen-2**

****

**Screen-3**

****

**TestGame3 – screenshots**

**Screen-1**

****

**Screen-2**

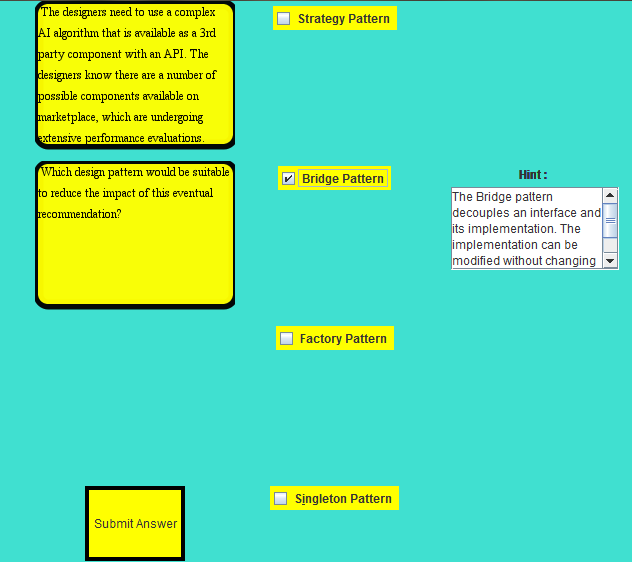
****

**Screen-3**

****

**TestGame4-Screenshots**

**Screen-1**

****

**Screen-2**

****

**Screen-3**

****

## References

1. JAXB - <http://docs.oracle.com/javase/tutorial/jaxb/intro/arch.html>
2. JAXB - <http://jaxb.java.net/tutorial/section_1_1-Introduction.html>
3. Java Coding Standards -<http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html>
4. Timing FrameWork Animation - <http://www.infoq.com/resource/articles/filthy-rich-clients/en/resources/FilthyRichClients14_InfoQ.pdf>