Florida International University
School of Computing and Information Sciences

CIS 4911 - Senior Capstone Project
Software Engineering Focus

Final Deliverable

Touch-midAir-Motion Gesture Input Framework
Team # X

Team Members
Ph.D. Francisco Ortega
Alberto Carrillo
David Escobar

Product Owner: Ph.D. Francisco Ortega
Instructor: Masoud Sadjadi

Abstract

In this document, we explore the design of the improved TAMGeF framework with the expanded Visualizers, LeapMotion Visualizer, Kinect Visualizer, and the TAMGeF API which acts as a graphics library for all of the current TAMGeF Visualizers. The sections to follow cover the system design of the visualizers and API, along with a description of the architectural patterns and design patterns that were chosen. Diagrams complement the described models and designs to aid in the understanding of the choices made. The testing methodologies were recorded and the important tests have been elaborated for the benefit of future developers.

Furthermore, a breakdown of the user stories and tasks worked on for the Visualizers and API is described in detail. This information is accompanied with details of the hardware and software used. Great emphasis was made on the work done during the sprints, as well as the way work was divided for each sprint. The team's experiences. both good and bad, have been detailed.

Table of Contents

INTRODUCTION

This project will form the basis for an I/O device management framework, designed for developers interested in developing applications for devices outside of the standard mouse and keyboard. Our purpose will be to generalize the existing visualizer to work with any kind of input device by the creation of a generalized API. Furthermore, we will display the API functionality by expanding the framework to include two new visualizers, LeapMotion visualizer and Kinect visualizer.

Current System

The current system offers a simple touch-screen visualizer, and a gyroscope visualizer, which are applications that receive input and relay the information visually to the user, either by displaying it on-screen or printing the data in text. The visualizers are designed for developers interested in developing applications for those input devices, and will be useful in gathering information on how the user interacts with the input devices.

The previous system's visualizer was capable of a few things: drawing the touched location on screen, recording the touch information to a JSON file, and replaying recorded touch-screen actions to the user. The visualizer also had extra features, such as changing the size of the brush during drawing, and displaying a console that gives immediate feedback on the actions being detected, it also provided algorithmic visualization of the data, such as minimum spanning tree and shortest hamiltonian path. The developers would be able to use this application to draw gestures in their touch-screen device and view the recorded results, allowing them to analyze this data for gesture processing.

The current system also includes a Gyroscope visualizer aimed for developers. The gyroscope visualizer provides developers with 3D visualization of the device and it also has three gauges: pitch indicator, accelerometer, and compass. Further functionality aside from the data provided by the three gauges and 3D representation of the device is non-existent.

The current system's visualizer also has no framework available to it. This means that the whole visualizers are stand-alone. There are many more limitations to the current system, starting with the constraint that it only serves for two input devices. The scope of the system is currently limited to analyzing and displaying data from these two input devices, and it does not provide any general way of expanding the amount of input devices that can be visualized. .

Purpose of New System

The purpose of the new system is not only to provide developers with an extended range of visualizers, such as the LeapMotion visualizer and Kinect visualizer. It will also provide a generalized API that will deal with abstracting input devices.

In more detail, developers will be able to use the LeapMotion visualizer, that will be powered by the API, to connect to the LeapMotion sensor in order to visualize the input data. Similarly, developers will be able to use the Kinect visualizer, powered by the API, to connect to the Kinect sensor and immediately visualize the input data gathered from the Kinect sensor.

Furthermore, the new system will provide a generalized API that any developer can use to immediately connect to any input device and visualize it in any way they choose by making use and compounding the primitive shapes that the TAMGeF API provides.

User Stories

This section will cover all of the user stories that were worked upon . The main goal of these user stories is to create a generalized API to visualize input devices and lastly to expand the framework to include the Kinect and LeapMotion input devices. The majority of work shown below reflects the development of this system.

Implemented User Stories

User Story # 409 - Read C++ STL book

As a team developer, I need to understand STL containers to work towards a generalization for the visualization project.

User Story # 410 - Understand advanced STL iterators, functions, and algorithms

As a team developer, I need to understand advanced STL iterators, functions, and algorithms to work towards a generalization for the visualization project

User Story # 411 - C++ Training 1

As a team developer, I need to understand C++ to work towards a generalization for the visualization project

User Story # 421 - Read C++ STL Book(David)

As a team developer, I need to understand STL containers to work towards a generalization for the visualization project.

User Story # 420 - Understand LeapMotion SDK

As a team developer, I need to understand the Leap Motion SDK, which will help towards a generalization for the visualization project.

User Story # 422 - Read Qt Book or Web
As a team developer, I need to understand the basics of QT to work towards a generalization for the visualization project.

User Story # 419 - Read C++ Tour Book
As a team developer, I need to understand C++ to work towards a generalization for the visualization project.

User Story # 412 - Understand Previous QT Realease Application (Multitouch)
As a team developer, I need to understand the previous version developed by summer 15 students, which will help towards a generalization for the visualization project

User Story # 423 - Fix OpenGL Bug
Fix OpenGL bug in TAMVisualizer

User Story # 413 - Read C++ API Design Book
As a team developer, I need to understand C++ API design to aid the decision making for the visualization API.

User Story # 416 - Meeting(Alberto)
As a team developer, I need to plan the sprint with my product owner.

User Story # 417 - Meeting(David)
As a team developer, I need to plan the sprint with my product owner.

User Story # 424 - Meeting with Alain(OpenGL Migration) Part I
As a team developer, I need to meet with Alain in order to migrate the current OpenGL implementation (fixed pipeline) to a more updated version (programmable pipeline).

User Story # 425 - Meeting with Alain(OpenGL Migration) Part II

As a team developer, I need to meet with Alain in order to migrate the current OpenGL implementation (fixed pipeline) to a more updated version (programmable pipeline).

User Story # 426 - Read OpenGL Documentation
As a team developer, I need to familiarize myself with OpenGL 4 (or higher) in order to successfully migrate the current implementation to a more updated one

User Story # 427 - Finish OpenGL Migration
As a team developer, I need to migrate the current OpenGL implementation to a more updated one

User Story # 428 - Meeting for Design of Visualizing API
As a team developer, I need to meet with Dr. Ortega in order to successfully design an API for the generalization idea for the visualizer.

User Story # 429 - Design Visualizing API
As a team developer, I will design the visualizing API using diagrams and/or code.

User Story # 431 - Create a Leap Motion data application (Part A)
As a team developer, I need to implement a GUI program that provides information (raw data) using the Leap Motion in order to understand the functioning of the device.

User Story # 432 - C++ Training II
As a team developer, I need to attend to the second Part of the C++ Training in order to get the skills to work in the project.

User Story # 433 - Understand Leap Motion API (Part B)
As a team developer, I need to understand and familiarize with the Leap Motion API .

User Story # 434 - Create a Leap Motion data application (Part B)

As a team developer, I need to implement a GUI program that provides information (raw data) using the Leap Motion in order to understand the functioning of the device.

User Story # 434 - Create a Leap Motion data application (Part B)
As a team developer, I need to implement a GUI program that provides information (raw data) using the Leap Motion in order to understand the functioning of the device.

User Story # 436 - Create documentation Fall 2015 (Part A)
As a team developer, I need to understand the functioning of the system to make changes later

User Story # 438 - Implementation of visualizing API Part A
As a team developer, I need to implement the visualizing API

User Story # 439 - Implementation of visualizing API Part B
As a team developer, I need to implement the visualizing API

User Story # 440 - Implementation of visualizing API Part C
As a team developer, I need to implement the visualizing API

User Story # 441 - Meeting for Visualizing API
Meeting with Product Owner

User Story # 442 - Create Design Documentation for Visualizing API
As a team developer, I need to create documentation for the visualizing API to start visualization API implementation and discuss design on meeting.

User Story # 443 - Create Design Documentation for Visualizing API Part 2
As a team developer, I need to create documentation for the visualizing API to start visualization API implementation and discuss design on meeting.

User Story # 444 - Create Leap Motion Application on Windows Part A

As a team developer, I need to create a leap motion application and setup development environment on windows so we can integrate input device to current Qt application.

User Story # 445 - Create Leap Motion Application on Windows Part B
As a team developer, I need to create a leap motion application and setup development environment on windows so we can integrate input device to current Qt application.

User Story # 446 - Create Leap Motion Application on Windows Part C
As a team developer, I need to create a leap motion application and setup development environment on windows so we can integrate input device to current Qt application.

User Story # 447 - Create Leap Motion Application Documentation
As a team developer, I need to create Leap Motion documentation for the Qt application we are working on.

User Story # 448 - Meeting with Francisco

As team developer, I need to meet with the product owner to continue working on the Leap Motion application.

User Story # 449 - Setup Visualization App from Github and Design Integration for the App

As a team developer, I need to setup the visualization app on a windows computer to understand how we are going to integrate leap motion with the current application.

User Story # 450 - Create GitHub Documentation
As a team developer, it is important to document how to upload files to GitHub repositories for future reference.

User Story # 451 - Meeting with Dr. Ortega
As a team developer, it is important to have a 2 hours meeting with Dr. Ortega to discuss mid-layer design

User Story # 452 - Remove Qt Dependencies in GL Layer
As a team developer, I want to remove all Qt dependencies in GL layer in order for the API to work in any OpenGL context

User Story # 453 - Understand Leap Motion
As a team developer, it is important to understand the Leap Motion device and see how it adapts with the visualization API.

User Story # 454 - Start Mid-layer design
As a team developer, I want to start the mid-layer design of the visualization API .

User Story # 455 - Create Leap Motion documentation
As a team developer I need to create a document explaining the functionality and data from leap motion.

User Story # 456 - Push code into GitHub
As a team developer I need to Push the code I created into our GitHub Project Page

User Story # 457 - Create Kinect Application Part I
As a team developer I need to understand how to extract and Implement data from Kinect Sensor and use it as needed in the application.

User Story # 458 - Create Kinect Application Part II
As a team developer I need to understand how to extract and Implement data from Kinect Sensor and use it as needed in the application.

User Story # 459 - Create Kinect Application Part III
As a team developer I need to understand how to extract and Implement data from Kinect Sensor and use it as needed in the application.

User Story # 460 - Create Documentation for Kinect
As a team developer I need to Create a document explaining the functionality and data from Kinect.

User Story # 461 - Push code into GitHub
As a team developer I need to Push the code I created into our GitHub Project Page

User Story # 462- Introduce Shape compounding to API
As a team developer, I want to introduce shape compounding to the API.

User Story # 463 - Develop Connections for Shapes in the API
As a team developer, it is important for me to introduce a way for developers to connect their shapes

User Story # 464 - Leap Motion running with API
As a team developer, it is important to test the API with the Leap Motion device.

User Story # 466 - Implement Kinect Skeleton
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display it as text in Console application

User Story # 467 - Implement Kinect Skeleton II
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display it as text in Console application

User Story # 468 - Implement Kinect Skeleton III
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display it as text in Console application

User Story # 469 - Kinect Documentation
Write Kinect Documentation for application

User Story # 470 - Meeting with Francisco
As a team developer, I need to meet with the product owner to continue working on the Kinect sensor application.

User Story # 471 - Kinect Skeleton API
As a team developer I need to be able to understand the Skeleton API of Kinect to implement it in my own application.

User Story # 472 - Unit Testing for API

As a team developer, it is important for me to develop unit testing for the API.

User Story # 473 - Introduce 3D Shapes API
As a team developer, it is important for me to introduce some 3D Shapes into the API

User Story # 474 - Improve API with David's Feedback
As a team developer, it is important for me to have the best API we can have. As such, it is important that I improve the API according to David's feedback.

User Story # 475 - Implement Kinect Skeleton Using API
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display its Coordinates as a point using the API.

User Story # 476 - Implement Kinect Skeleton Using API II
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display its Coordinates as a point using the API.

User Story # 477 - Implement Kinect Skeleton Using API III
As a team developer I need to be able to pull all skeleton data from Kinect Sensor v2 and display its Coordinates as a point using the API.

User Story # 478 - Give feedback about the API to Alberto
As a team developer I need to give feedback to Alberto in order to improve the API.

User Story # 479 - Improve API Performance
As a team developer, it is important for me to improve the API performance.

User Story # 480 - Fix API Circle Window Size Bug

As a team developer, it is important for me to fix the circle bug for the API to work at its best

User Story # 481 - Work on Documentation
As a team developer, it is important for me to work on documentation so that new developers can have a good reference for the future.

User Story # 482 - Add color selection for 3D Shapes
As a team developer, it is important for me to implement color selection for 3D Shapes so that the API is fully customizable.

User Story # 483 - Work with David on the improvement of the QtLeapGL Application
As a team developer, it is important for me to improve the functionality of the QtLeapGL application so that it can visually represent more features.

User Story # 484 - Improve Kinect application using the API
As a team developer I need to work in the Kinect application using the api in order to improve performance

User Story # 485 - Work in documentation
As a team developer I need to create the right documentation for further use

User Story # 486 - Work with Alberto on the improvement of the QtLeapGL Application
As a team developer, it is important for me to improve the functionality of the QtLeapGL application so that it can visually represent more features.

Pending User Stories

No user stories that are pending

PROJECT PLAN

Hardware and software resources are used to help develop the current TAM system. The Hardware and Software section describes the individual hardware and software components and provides a brief explanation as to why the respective hardware or software was chosen. The section thereafter details the work that has been accomplished in each of the sprints. The sprints are divided into several user stories that describe the completed tasks, which are then further divided into subtasks. Each user story in the section also includes the acceptance criteria (the specifications that had to be met in order to declare the task complete), and the modeling (a visual representation of the user story's functionality).

Hardware and Software Resources

Hardware Resources

- **Macbook Pro, Early 2011**
    - Using the Macbook Pro, arbitrarily, for David to program the Kinect Visualizer
- **Acer 21" Touch Screen Display**
    - The touch screen is necessary to test the visualizer application. The visualizer is meant to be used by touch screen devices, and so this touch screen provides the best means of testing the application.
- **LeapMotion Input Device**
    - One of the main input devices for this iteration of the project. It includes a sensor that tracks the position of hands and fingers
- **Kinect Input Device**
    - One of the main input devices for this iteration of the project. It includes a sensor that tracks the position of each bone in the human body.
- **Custom built computer**
    - Custom built computer with high end CPU and videocard for Alberto to program the API and LeapMotion Visualizer.

Software Resources

- **Windows 7**
    - Necessary for installing Visual Studio 2013
- **Visual Studio 2013**
    - Necessary IDE as it was used in previous versions to program the visualizers.
- **C++ Programming Language**
    - The language of choice by Dr. Francisco Ortega. It was used in the previous iteration of the project and as such is the language for the new system. It was chosen due to its efficiency for computer graphics.
- **Qt Framework**
    - Used for its capacity to interact with touch screens and its ease of use when creating visual windows.
- **OpenGL Framework**
    - Used to power the API render features such as 2D shapes and 3D shapes
- **GLM API**
    - Used for the TAMGeF API to provide support for matrices and varied math functions.
- **LeapMotion SDK**
    - Used to interact with the LeapMotion device in order to program the LeapMotion visualizer.
- **Kinect SDK**
    - Used to interact with the Kinect device in order to program the Kinect visualizer

Sprints Plan

Sprint 1
(08/28/2015 - 09/11/2015)

User Story # 412 - Understand Previous QT Release Application (Multitouch)

Tasks
- Download and run application
- Understand Basics of Qt
- Understand Qt Multithreading
- Understand Multi-touch input interaction

Acceptance Criteria
- Application downloaded and running
- Basics of Qt understood
- Qt Multithreading understood
- Multi-touch input interaction understood

User Story # 420 - Understand LeapMotion SDK

Tasks
- Understand and run LeapMotion Samples
- Understand basics of LeapMotion SDK

Acceptance Criteria
- Leapmotion samples understood and run
- LeapMotion SDK basics understood

User Story # 410 - Understand advanced STL iterators, functions, and algorithm

Tasks
- Read Chapter 9 C++ STL Book
- Read Chapter 10 C++ STL Book
- Read Chapter 11 C++ STL Book

Acceptance Criteria

- Chapter 9 read
- Chapter 10 read
- Chapter 11 read

User Story # 423 - Fix OpenGL Bug

Tasks

- Fix OpenGL Bug

Acceptance Criteria

- TAMVisualizer must be able to draw to screen

User Story # 411 - C++ Training 1

Tasks

- Attend C++ training meeting

Acceptance Criteria

- Meeting attended

User Story # 422 - Read Qt book or web

Tasks

- Read first 5 chapters of book on Qt

Acceptance Criteria

- First five chapters read

User Story # 421 - Read C++ STL Book(David)
Tasks
- Read Chapter 7
- Read Chapter 8
- Read Chapter 9

Acceptance Criteria
- Chapter 7 read
- Chapter 8 read
- Chapter 9 read

User Story # 419 - Read C++ Tour Book
Tasks
- Read C++ Tour Book

Acceptance Criteria
- C++ Tour Book read

User Story # 413 - Read C++ API Design Book
Tasks
- Read Chapter 2 of Modern API Design
- Read Chapter 3 of Modern API Design
- Read Chapter 5 of Modern API Design

Acceptance Criteria
- Chapter 2 read
- Chapter 3 read
- Chapter 5 read

User Story # 417 - Meeting(David)
Tasks
- Meet with Product Owner

Acceptance Criteria
- Attend 2 hour meeting

User Story # 416 - Meeting(Alberto)
Tasks
- Meet with Product Owner

Acceptance Criteria
- Attend 2 hour meeting

Sprint 2
(09/11/2015 - 09/25/2015)

User Story # 427 - Finish OpenGL Migration
Tasks
- Migrate Visualizer to newer OpenGL version

Acceptance Criteria
- Visualizer migrated to newer OpenGL version

User Story # 437 - Attend design of API meeting
Tasks
- Meet with Dr. Ortega

Acceptance Criteria
- Met with Dr. Ortega

User Story # 424 - Meeting with Alain(OpenGL Migration) Part I
Tasks
- Meet with Alain to review OpenGL programmable pipeline

Acceptance Criteria
- Met with Alain

User Story # 425 - Meeting with Alain(OpenGL Migration) Part II
Tasks
- Meet with Alain to review OpenGL programmable pipeline

Acceptance Criteria
- Met with Alain

User Story # 426 - Read OpenGL Documentation
Tasks
- Read documentation on web

Acceptance Criteria
- Documentation read

User Story # 428 - Meeting for Design of Visualizing API
Tasks
- Meet with Dr. Ortega

Acceptance Criteria
- Met with Dr. Ortega

User Story # 429 - Design Visualizing API
Tasks
- Start design of visualizing API by providing diagrams and/or code

Acceptance Criteria
- Visualizing API design provided

User Story # 432 - C++ Training II

Tasks
- Attend C++ training meeting

Acceptance Criteria
- Meeting attended

User Story # 431 - Create a Leap Motion data application (Part A)
Tasks
- Read documentation on web
- Display data on a GUI

Acceptance Criteria
- GUI has been provided

User Story # 433 - Create a Leap Motion data application (Part B)
Tasks
- Read documentation on web
- Display data on a GUI

Acceptance Criteria
- GUI has been provided

User Story # 434 - Create a Leap Motion data application (Part C)
Tasks
- Read documentation on web
- Display data on a GUI

Acceptance Criteria
- GUI has been provided

User Story # 436 - Create documentation Fall 2015 (Part A)
Tasks
- Start documentation for initial API generalization

Acceptance Criteria
- Documentation provided

Sprint 3
(09/25/2015 - 10/09/2015)

User Story # 438 - Implementation of visualizing API Part A
Tasks
- The API GL Layer needs to be implemented

Acceptance Criteria
- API GL Layer implemented

Modeling
- Figure 20
- Figure 21

User Story # 439 - Implementation of visualizing API Part A
Tasks
- The API GL Layer needs to be implemented

Acceptance Criteria
- API GL Layer implemented

Modeling
- Figure 20
- Figure 21

User Story # 440 - Implementation of visualizing API Part A
Tasks
- The API GL Layer needs to be implemented

Acceptance Criteria
- API GL Layer implemented

Modeling
- Figure 20
- Figure 21

User Story # 441 - Meeting for Visualizing API
Tasks
- Attend Meeting 1
- Attend Meeting 2

Acceptance Criteria

- Meetings attended

User Story # 442 - Create Design Documentation for Visualizing API
Tasks

- Create design documentation to be used for implementation and discussed on the meeting

Acceptance Criteria

- Documentation provided
-

User Story # 443 - Create Design Documentation for Visualizing API Part II
Tasks

- Create design documentation to be used for implementation and discussed on the meeting

Acceptance Criteria

- Documentation provided

User Story # 444 - Create Leap Motion Application on Windows Part A
Tasks

- Create a Leap Motion application for windows

Acceptance Criteria

- Application uses visual studio
- Application uses leap motion
- Application uses Qt framework

User Story # 445 - Create Leap Motion Application on Windows Part B
Tasks

- Create a Leap Motion application for windows

Acceptance Criteria

- Application uses visual studio
- Application uses leap motion
- Application uses Qt framework

User Story # 446 - Create Leap Motion Application on Windows Part C
Tasks

- Create a Leap Motion application for windows

Acceptance Criteria

- Application uses visual studio

- Application uses leap motion
- Application uses Qt framework

User Story # 447 - Create Leap Motion Application Documentation
Tasks
- Create diagrams for the LeapMotion visualizer

Acceptance Criteria
- Class diagrams provided
- Sequence diagrams provided
- Internal documentation provided

User Story # 448 - Meeting with Francisco
Tasks
- Meet with Francisco

Acceptance Criteria
- Met with Francisco

User Story # 449 - Setup Visualization App from Github and Design Integration for the App
Tasks
- Setup visualization applications from github into development environment

Acceptance Criteria
- Setup environment
- Understand code
- Propose way to integrate LeapMotion

Sprint 4

(10/09/2015 - 10/23/2015)

User Story # 454 - Start Mid-Layer Design

Tasks

- Start Mid-Layer design of the TAMGeF API

Acceptance Criteria

- Mid-Layer design started
- API drawing shapes

Modeling

- Figure 9
- Figure 12
- Figure 16
- Figure 18

User Story # 451 - Meeting with Dr. Ortega

Tasks

- Meet with Dr. Ortega

Acceptance Criteria

- Met with Dr. Ortega

User Story # 452 - Remove Qt Dependencies in GL Layer

Tasks

- Remove any Qt dependency from the API GL Layer

Acceptance Criteria

- Dependencies removed

User Story # 453 - Understand LeapMotion

Tasks

- Read LeapMotion documentation to understand the device

Acceptance Criteria

- LeapMotion device understood

User Story # 455 - Create LeapMotion documentation

Tasks

- Provide documentation for the LeapMotion

Acceptance Criteria

- Detailed documentation on how to read data from LeapMotion

User Story # 456 - Push code into GitHub

Tasks

- Push current code to GitHub

Acceptance Criteria

- Code has been pushed

User Story # 457 - Create Kinect Application Part I

Tasks

- Understand the Kinect API for further development

Acceptance Criteria

- API Understood

User Story # 458 - Create Kinect Application Part II

Tasks

- Understand the Kinect API for further development

Acceptance Criteria

- API Understood

User Story # 459 - Create Kinect Application Part III

Tasks

- Understand the Kinect API for further development

Acceptance Criteria

- API Understood

User Story # 460 - Create documentation for Kinect

Tasks

- Create detailed documentation explaining all aspects of Kinect

Acceptance Criteria

- Documentation provided

User Story # 450 - Create GitHub Documentation

Tasks

- Create documentation for future developers to know how to use GitHub

Acceptance Criteria

- GitHub usage guide provided

Sprint 5
(10/23/2015 - 11/06/2015)

User Story # 462 - Introduce Shape Compounding to the API

Tasks

- Introduce Shape compounding to the API so that developers can build new shapes from existing shapes

Acceptance Criteria

- Shapes can be compounded using the API

Modeling

- Figure 13
- Figure 17

User Story # 463 - Develop connections for shapes in the API

Tasks

- Shapes must be easily connected by a line using the API

Acceptance Criteria

- Connection has been introduced to the API

User Story # 464 - LeapMotion running with API

Tasks

- Develop a LeapMotion application that runs using the API

Acceptance Criteria

- LeapMotion visualizer running using the API

Modeling

- Figure 30

User Story # 466 - Implement Kinect Skeleton

Tasks

- Extract all joint data from the Kinect sensor and display it in textual form

Acceptance Criteria

- Application runs displaying joints position

User Story # 467 - Implement Kinect Skeleton II

Tasks

- Extract all joint data from the Kinect sensor and display it in textual form

Acceptance Criteria

- Application runs displaying joints position

User Story # 468 - Implement Kinect Skeleton III

Tasks

- Extract all joint data from the Kinect sensor and display it in textual form

Acceptance Criteria

- Application runs displaying joints position

User Story # 469 - Kinect documentation

Tasks

- Write documentation for Kinect application

Acceptance Criteria

- Documentation provided

User Story # 470 - Meeting with Francisco

Tasks

- Meet with Product Owner

Acceptance Criteria

- Met with Product Owner

User Story # 471 - Kinect Skeleton API

Tasks

- Understand Skeleton API

Acceptance Criteria

- Skeleton API understood

Sprint 6
(11/06/2015 - 11/20/2015)

User Story # 472 - Unit Testing for API
Tasks
- Develop Unit Testing for the API

Acceptance Criteria
- Unit Testing developed for the API

User Story # 473 - Introduce 3D Shapes for the API
Tasks
- Introduce 3D Shapes to the API such as a cube and pyramid

Acceptance Criteria
- 3D Shapes introduced

Modeling
- Figure 10
- Figure 11
- Figure 14
- Figure 15

User Story # 475 - Implement Kinect Skeleton using API
Tasks
- Implement a skeleton of the human body drawn by using the TAMGeF API

Acceptance Criteria
- Kinect visualizer running with the API and properly displaying the skeleton

Modeling
- Figure 34

User Story # 476 - Implement Kinect Skeleton using API II
Tasks
- Implement a skeleton of the human body drawn by using the TAMGeF API

Acceptance Criteria
- Kinect visualizer running with the API and properly displaying the skeleton

Modeling
- Figure 34

User Story # 477 - Implement Kinect Skeleton using API III

Tasks

- Implement a skeleton of the human body drawn by using the TAMGeF API

Acceptance Criteria

- Kinect visualizer running with the API and properly displaying the skeleton

Modeling

- Figure 34

User Story # 478 - Give Feedback About API to Alberto

Tasks

- Use the API and give feedback to Alberto based on my experiences

Acceptance Criteria

- Feedback has been provided to Alberto

Sprint 7

(11/20/2015 - 12/04/2015)

User Story # 479 - Improve API Performance

Tasks

- Improve the graphical performance of the API

Acceptance Criteria

- API performance increased

User Story # 480 - Fix API Circle Window Size Bug

Tasks

- Implement a matrix for the Circle in order to maintain the aspect ratio independent of the window size

Acceptance Criteria

- Bug fixed

User Story # 481 - Work on Documentation

Tasks

- Work on project documentation such as class, sequence, use case diagrams

Acceptance Criteria

- Diagrams provided

User Story # 482 - Add Color Selection for 3D Shapes
Tasks
- Implement color selection for 3D Shapes on the TAMGeF API

Acceptance Criteria
- Color selection implemented

User Story # 483 - Work with David on Improvement of the QtLeapGl Application
Tasks
- Improve the LeapMotion visualizer so that it not only tracks the palm of a hand but also the fingers

Acceptance Criteria
- LeapMotion visualizer tracking palm and fingers

User Story # 486 - Work with Alberto on Improvement of the QtLeapGl Application
Tasks
- Improve the LeapMotion visualizer so that it not only tracks the palm of a hand but also the fingers

Acceptance Criteria
- LeapMotion visualizer tracking palm and fingers

User Story # 484 - Improve Kinect Application using the API
Tasks
- Improve the LeapMotion visualizer so that it not only tracks the palm of a hand but also the fingers

Acceptance Criteria
- LeapMotion visualizer tracking palm and fingers

User Story # 485 - Work in Documentation
Tasks
- Create Documentation such as class diagrams, sequence diagrams and use case diagrams for the Kinect visualizer

Acceptance Criteria
- Documentation provided

SYSTEM DESIGN

System design defines the strategies taken by the developers to construct the desired project. Here we break up our design goals into the architectural patterns used, the system and subsystem decomposition models, and the design patterns. These make up a large part of the planning that was necessary for putting this project on course. An in-depth look at the choice's for this semester's design are looked at further in the upcoming subsections.

Architectural Patterns

For both the LeapMotion and Kinect visualizers the Model-View-Controller pattern was used. The Model-View-Controller pattern was also used in the development of the TAMGeF API. For all of these visualizers the view and controller are merged because the user interface not only allows the display of data but also allows user to control the display of data.

The streaming input data and the rendering goes into model because for all systems they subscribe to the streaming input data.

System and Subsystem Decomposition

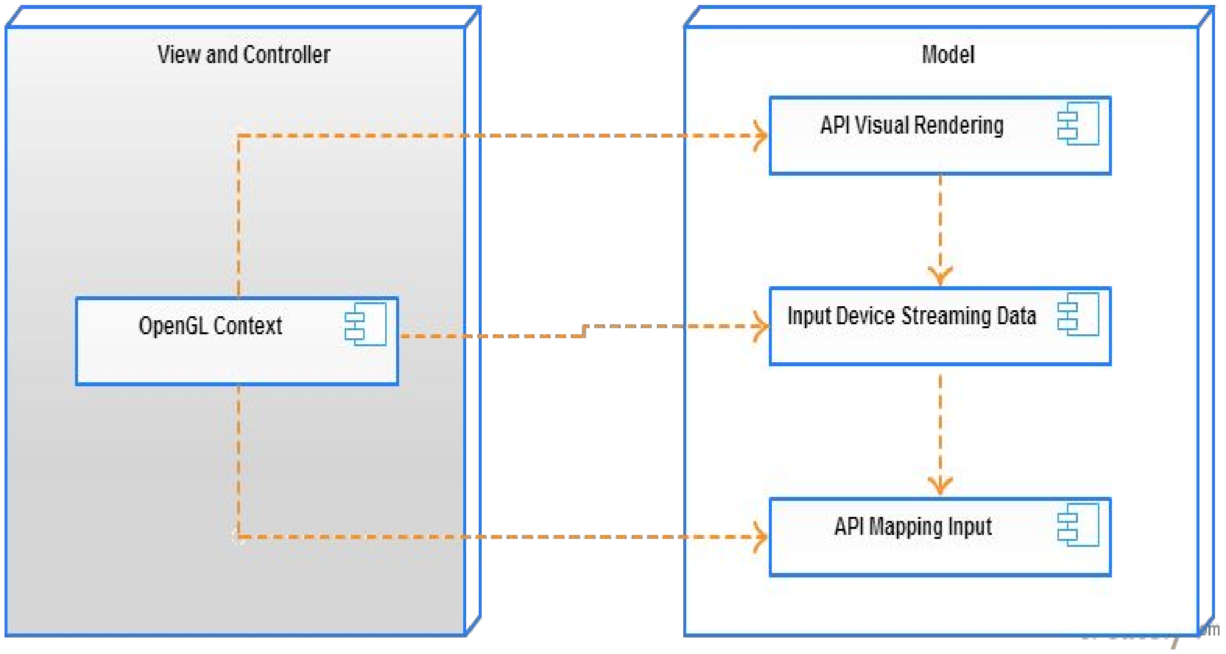

Figure 1 - Model-View-Controller Architecture

Above, in Figure 1, we have a component diagram detailing the subsystem decomposition in terms of the MVC architecture. For the sake of simplicity, the subsystems have been abstracted to encompass both the LeapMotion, and Kinect visualizers as well as the API.

The OpenGL Context subsystem covers all of the visual information that the user sees. This might be the Skeleton drawn by the Kinect or the hand drawn by the LeapMotion visualizer. The OpenGL Context depends on the Visual Rendering subsystem that is present in the visualizers and API. This subsystem is in charge of using either the TAMGeF API, which in turn uses OpenGL's API to render graphics that are seen on the OpenGL Context.
The API Visual Rendering requires data to display the information, and thus depends on the Input Device Streaming Data Subsystem, which tries to gather data from the devices, manipulate

the data, and emit the data to the appropriate rendering function in the appropriate class of the TAMGeF API.
Finally, the Input Device Streaming Data Subsystem depends on the API Mapping Input Subsystem to ensure that the data is streamed in a coordinate system that will allow proper rendering on the OpenGL Context.

Deployment Diagram

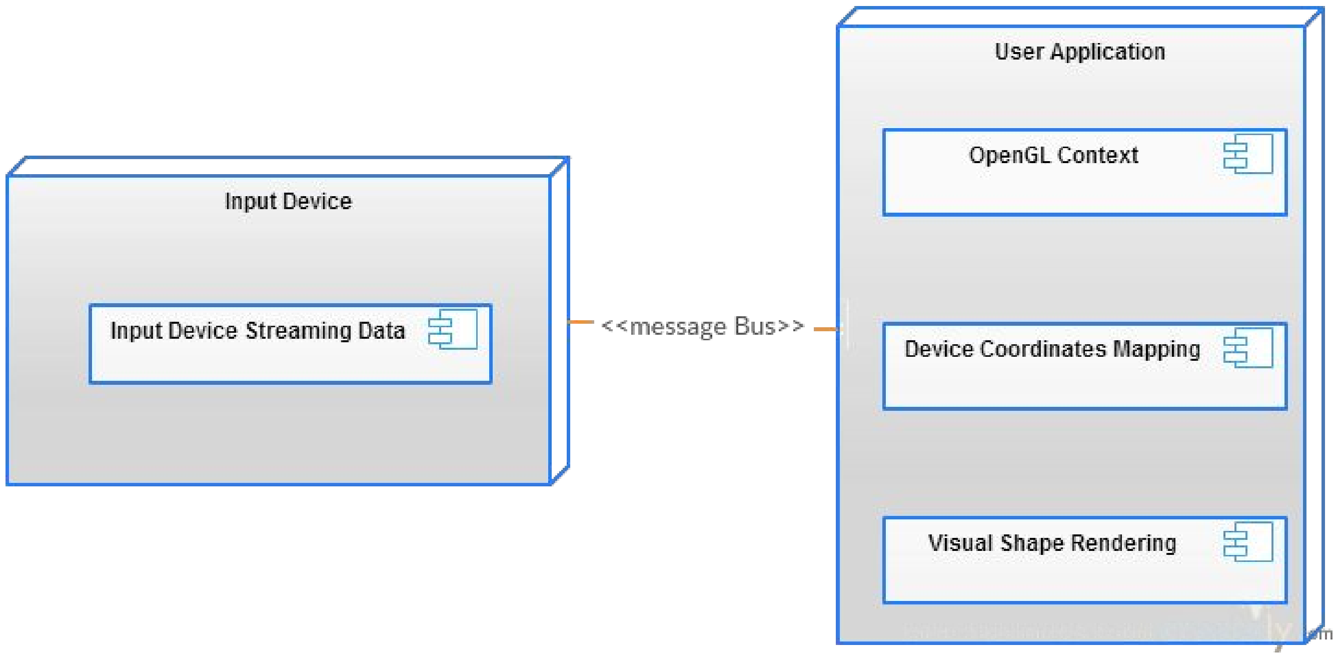

Figure 2 - Deployment Diagram

Design Patterns

Observer Pattern:
Similar as in previous iterations of the project an observer pattern was chosen in order to effectively use multithreading with the visualization code. The purpose of this pattern is that we are able to have a thread collecting input data and doing complex manipulation of the data, then

it emits the data to the TAMGeF API with another thread that has subscribed to the data collection thread. This is implemented on the LeapMotion visualizer and Kinect visualizer.

Command Pattern:
The Command Pattern was chosen as a way to separate the process between the UI thread and the inputhandler thread. With the Command pattern, the inputhandler thread can be fed data (from the LeapMotion input device or the Kinect input device) and perform the intense calculations that are needed to produce the results to display on screen, then send the commands to the UI thread on what to draw on the screen. Since the inputhandler will be sending the commands to the UI thread, it keeps the calculation process in a separate thread from the user interface process, allowing the UI to run seamlessly.

Facade Pattern:
For the TAMGeF API, the Facade Pattern was chosen as a way to provide a very easy to use interface for developers. With the Facade Pattern we abstract the complex and difficult to understand implementation of the TAMGeF API and instead provide a simple and easy to use interface that software developers can use to draw any kind of shapes. The Facade Pattern will be maintained for future iterations of the API.

SYSTEM VALIDATION

Both visualizers and the TAMGeF API used the Bottom-up approach to testing the software. The testing samples such as the LeapMotion data application and Kinect data application used a Big Bang testing approach .The project leverages a lot of external APIs and all of these have been tested, hence, our testing was minimal. The classes for the visualizers and TAMGeF API were written and tested individually, and as the sprints went on, we integrated the already tested components with the newer components and performed testing on those. We also included unit-tests for the TAMGeF API non-rendering functionality.

User Story # 423 - Fix OpenGL Bug

System Tests
- Test #001 - Run TAMVisualizer. Touch Screen. Verify all graphics are being drawn to screen.

User Story # 427 - Finish OpenGL Migration

System Tests
- Test #002 - Run TAMVisualizer. Touch Screen. Verify all graphics are being drawn to screen(Now using OpenGL Programmable pipeline) .

User Story # 431/434 - Create LeapMotion Data Applicaton

System Tests
- Test #003 - Run LeapMotion data application. Verify data is being displayed accurately.

User Story # 438/439/440 - Implementation of Visualizing API

System Tests
- Test #004 - Modify TAMVisualizer to use the automatic OpenGL context creation provided by the API. Run TAMVisualizer. Verify context is being created(VisualizerHandler class).

- Test #005 - Modify TAMVisualizer to use the automatic OpenGL context creation provided by the API. Run TAMVisualizer. Verify context can be resized(VisualizerHandler class).
- Test #006 - Modify TAMVisualizer to use the automatic OpenGL context creation provided by the API. Run TAMVisualizer. Verify context can display graphics(VisualizerHandler class).

User Story # 444/445/446 - Create LeapMotion Application on Windows
System Tests

- Test #007 - Run LeapMotion windows application. Verify fingertips are being drawn to screen.

User Story # 452 - Remove Qt Dependencies in GL Layer
System Tests

- Test #008 - Run Visualizer powered by API. Verify all drawing is working correctly.

User Story # 454 - Start Mid-Layer Design
System Tests

- Test #009 - Open Visualizer. Touch Screen. Verify Circles are being draw to screen on the appropriate coordinates(Test for Circle class of the API).
- Test #010 - Open Visualizer. Touch Screen. Verify Triangles are being draw to screen on the appropriate coordinates(Test for Triangle class of the API).
- Test #011 - Open Visualizer. Touch Screen. Verify Squares are being draw to screen on the appropriate coordinates(Test for Square class of the API).
- Test #012 - Open Visualizer. Touch Screen. Verify Right Triangles are being draw to screen on the appropriate coordinates(Test for Square class of the API).
- Test #013 - Open Visualizer. Touch Screen. Verify Lines are being draw to screen on the appropriate coordinates(Test for Line class of the API).

User Story # 462 - Introduce Shape Compounding to API
System Tests
- Test #014 - Open Visualizer. Touch Screen. Verify Right Triangle can successively be compounded in all directions(Test for Right Triangle class of the API).
- Test #015 - Open Visualizer. Touch Screen. Verify Line can successively be compounded in all directions(Test for Line class of the API).

User Story # 463 - Develop Connections for Shape in the API
System Tests
- Test #016 - Open Visualizer. Touch Screen. Verify two or more shapes can be successively connected by lines(Test for Line class of the API).

User Story # 457/458/459/466/467/468 - Create Kinnect Application
System Tests
- Test #017 - Open Kinnect data application. Verify the application can display the data gathered by the sensor in textual form.

User Story # 464 - LeapMotion running with API
System Tests
- Test #018 - Run LeapMotion visualizer. Verify the InputHandler class is gathering the right coordinates and passing them to the graphical interface.
- Test #019 - Run LeapMotion visualizer. Verify the hand is being drawn as intended(Hand class of the visualizer).

User Story # 472 - Unit Testing for API
System Tests
- Test #020 - InputMapper Construction mapping coordinates correctly.
- Test #021 - InputMapper setXCoordinate() setting coordinates correctly.
- Test #022 - InputMapper setYCoordinate() setting coordinates correctly.
- Test #023 - InputMapper setZCoordinate() setting coordinates correctly.
- Test #024 - InputMapper setWCoordinate() setting coordinates correctly.
- Test #025 - InputMapper getXCoordinate() getting coordinates correctly.
- Test #026 - InputMapper getYCoordinate() getting coordinates correctly.
- Test #027 - InputMapper getZCoordinate() getting coordinates correctly.
- Test #028 - InputMapper getWCoordinate() getting coordinates correctly.
- Test #029 - RightTriangle TOP compounding returning the correct offset.
- Test #030 - RightTriangle BOT compounding returning the correct offset.
- Test #031 - RightTriangle LEFT compounding returning the correct offset.

- Test #032 - RightTriangle RIGHT compounding returning the correct offset.
- Test #033 - Test Helpers pixels to gl coordinates conversion is correct.
- Test #034 - Test Helpers setScreenSize returning correct coordinates.
- Test #035 - Test Helpers LoadShaders loading shaders correctly.
- Test #036 - Test Helpers generateColor returning the expected colors.

User Story # 473 - Introduce 3D Shapes to API
System Tests
- Test #037 - Run Visualizer. Touch Screen. Verify a Cube is being displayed on the screen(Cube Class).
- Test #038 - Run Visualizer. Touch Screen. Verify a Cube can be compounded from all directions(Cube Class).
- Test #039 - Run Visualizer. Touch Screen. Verify a Pyramid is being displayed on the screen(Pyramid Class).
- Test #040 - Run Visualizer. Touch Screen. Verify a Pyramid can be compounded from all directions(Pyramid Class).
- Test #041 - Run Visualizer. Touch Screen. Verify a Pyramid can be rotated in all directions(Pyramid Class).

User Story # 475/476/477 - Implement Kinect Skeleton using API
System Tests
- Test #042 - Run Kinect Visualizer. Place body in front of sensor. Verify all bones are being drawn as lines(myBody class).
- Test #043 - Run Kinect Visualizer. Place body in front of sensor. Verify all lines are being drawn in the right position(InputHandler class).
- Test #044 - Run Kinect Visualizer. Place body in front of sensor. Verify all body movement can be handled correctly.

User Story # 480 - Fix API Circle Window Size Bug
System Tests
- Test #045 - Verify the Circle can be drawn on any screen size and keep its aspect ratio.

User Story # 480 - Add Color Selection for 3D Shapes
System Tests

- Test #046 - Run Visualizer. Touch screen. Verify Cube is displaying selected colors.
- Test #047 - Run Visualizer. Touch screen. Verify Pyramid is displaying selected colors.

User Story # 483/486 - Work with David/Alberto on the Improvement of the QtLeapGL Application
System Tests

- Test #048 - Run LeapMotion Visualizer. Place hand over LeapMotion. Verify the hand is being drawn as a combination of Circles, lines, and a square using the API.
- Test #049 - Run LeapMotion Visualizer. Place hand over LeapMotion. Verify the position of the fingers is correct.
- Test #050 - Run LeapMotion Visualizer. Place hand over LeapMotion. Verify the position of the palm is correct.
- Test #051 - Run LeapMotion Visualizer. Place hand over LeapMotion. Verify the hand and fingers can move correctly.

User Story # 484 - Improve Kinect Application using the API
System Tests

- Test #052 - Run Kinect Visualizer. Place body in front of sensor. Verify that circles are being drawn in every joint.

Glossary

- **TAM**: Short for “Touch-midAir-Motion”
- **Qt**: A framework widely used for developing applications with graphical user interfaces.
- **TAMGeF API**: The application programming interface developed in this project. Acts as our own graphics library.
- **Visualizer:** Refers to any graphical user interface that, running our TAMGeF API, graphically represents the data transmitted by an input device.

APPENDIX

Appendix A - UML Diagrams

Static UML Diagrams

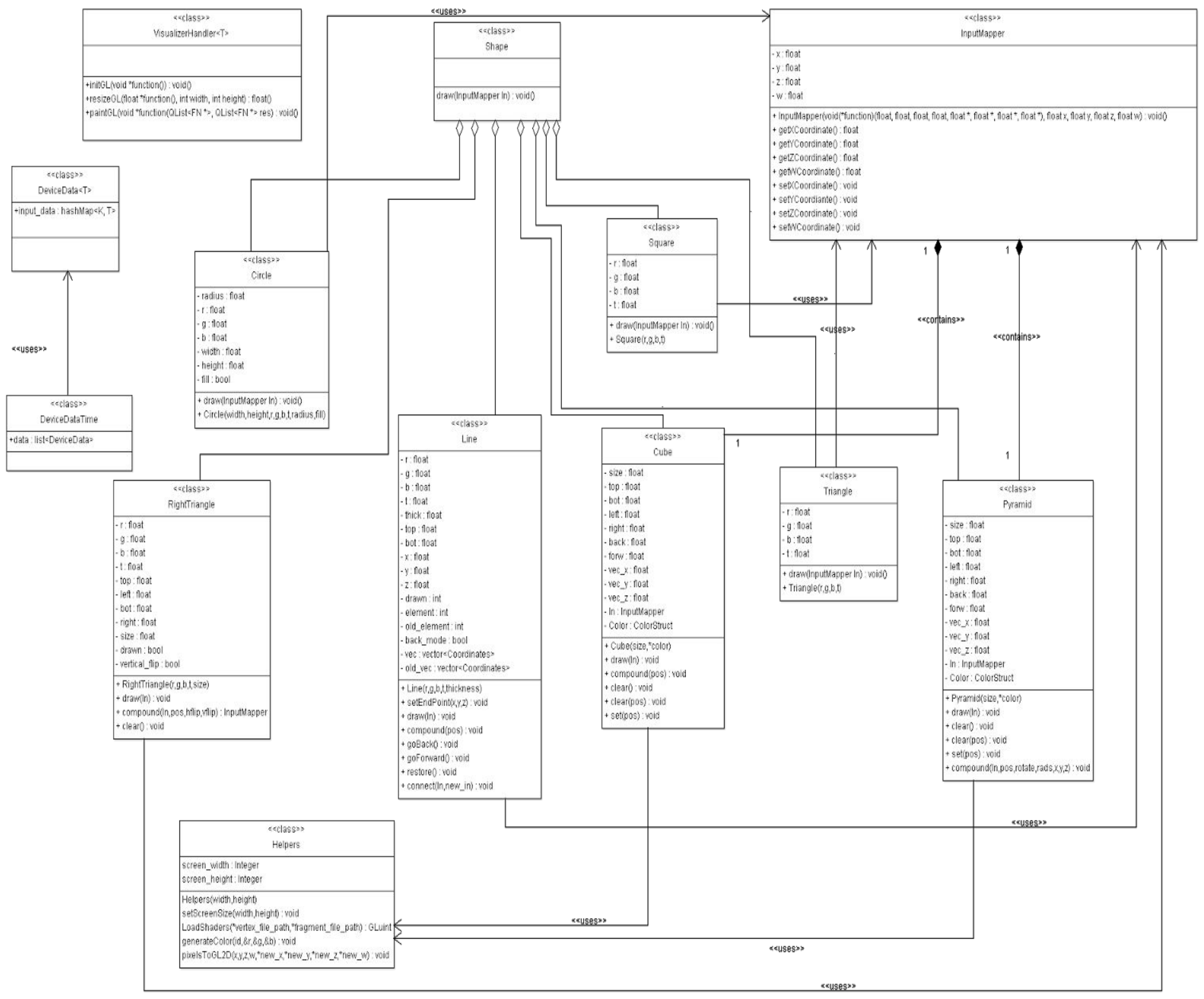

Figure 3 - API Class Diagram

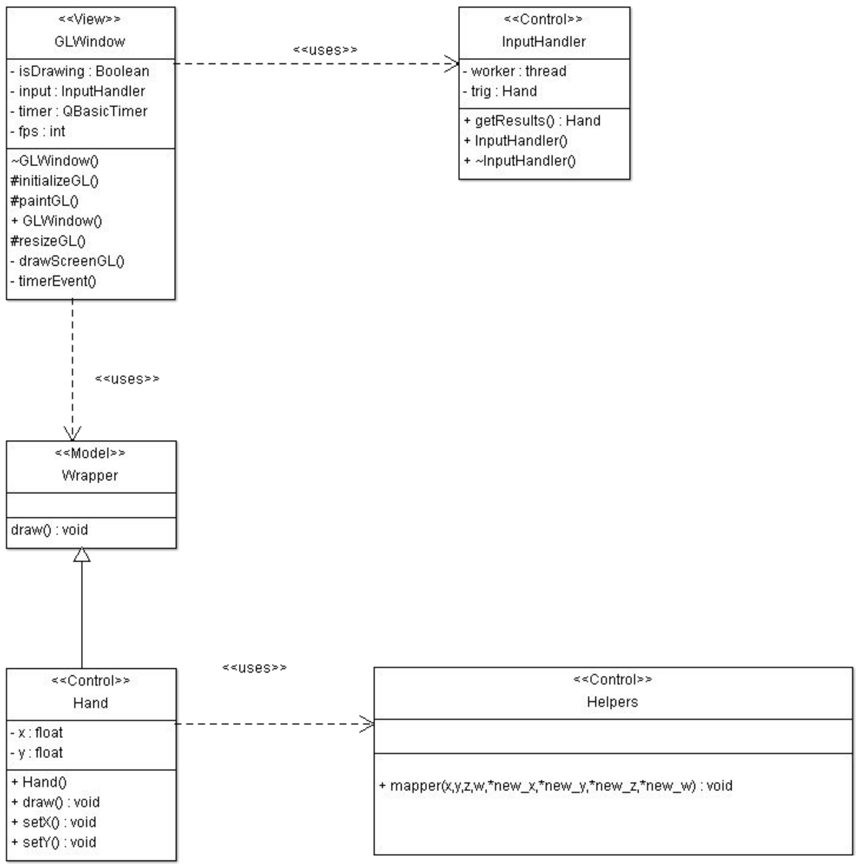

Figure 4 - LeapMotion Visualizer Class Diagram

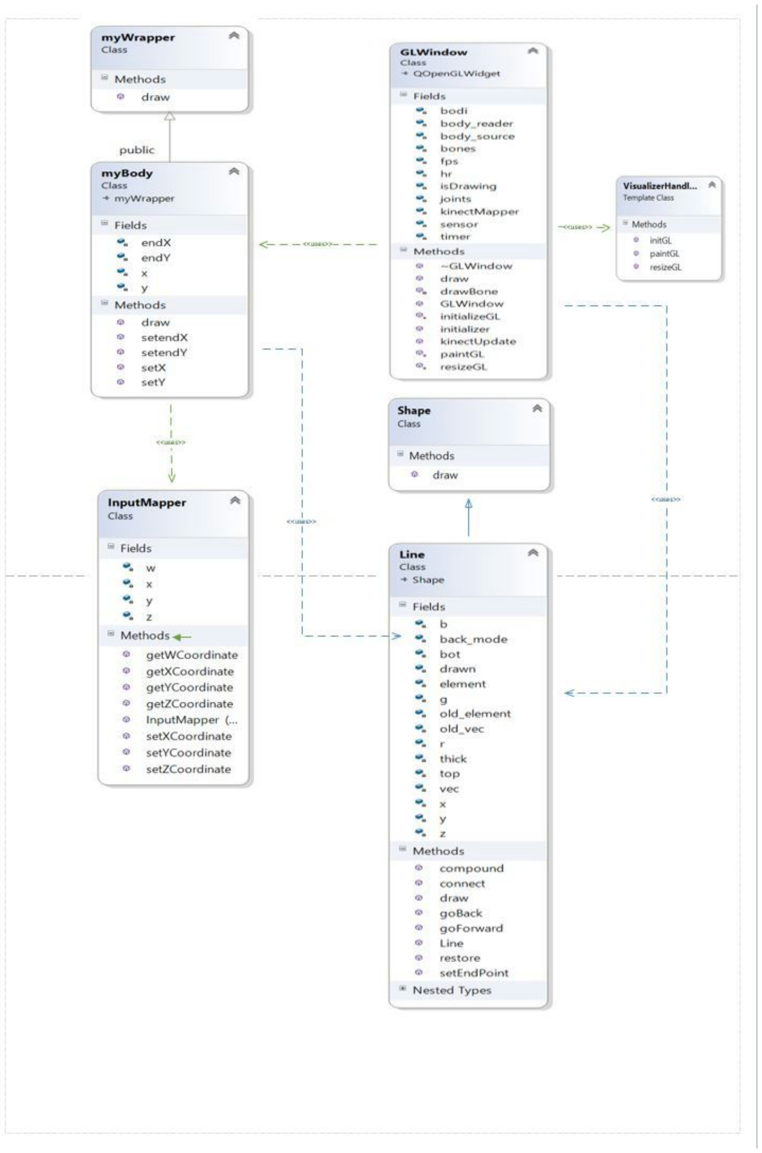

Figure 5 - Kinect Visualizer Class Diagram

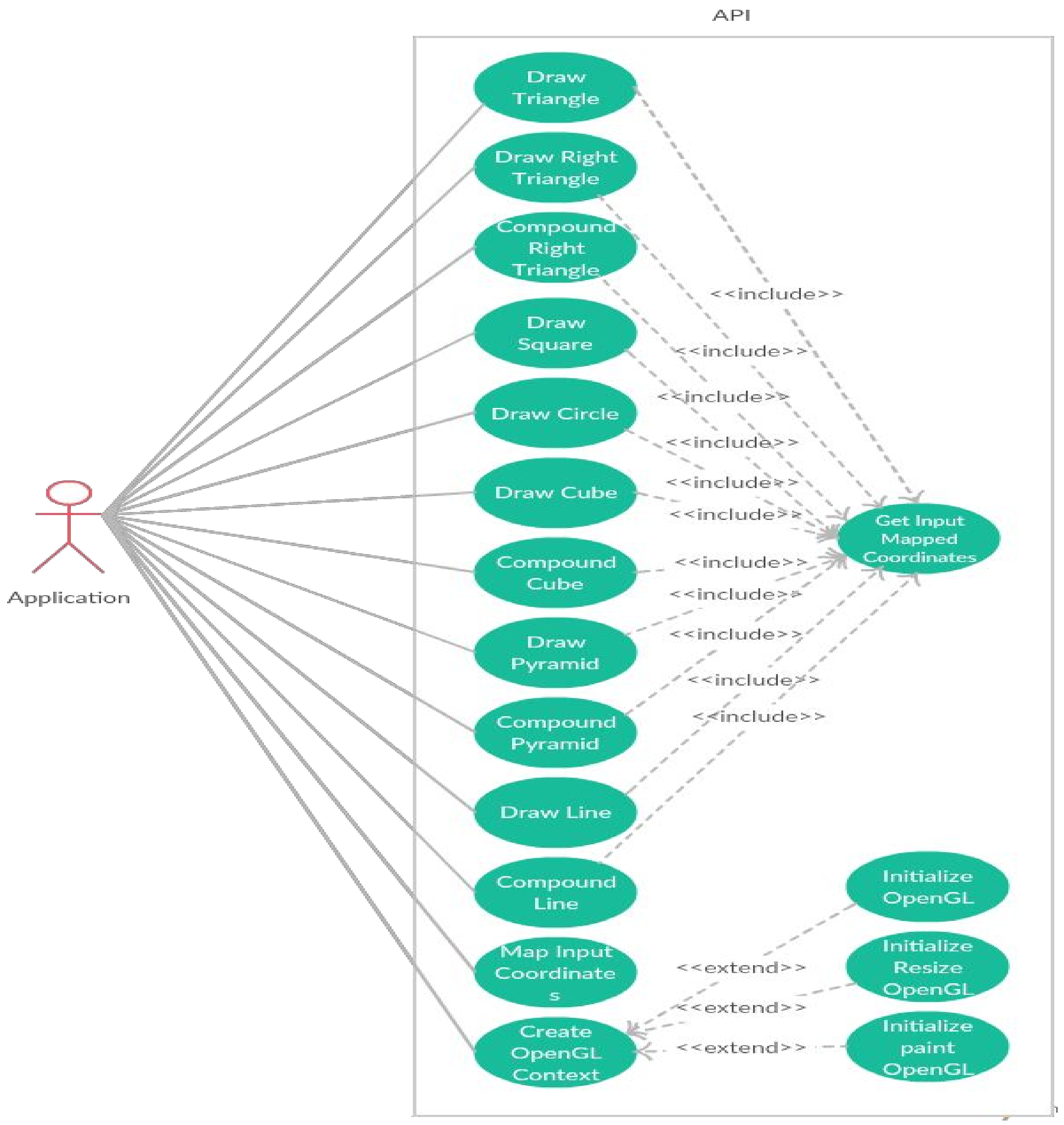

Figure 6 - API Use Case Diagram

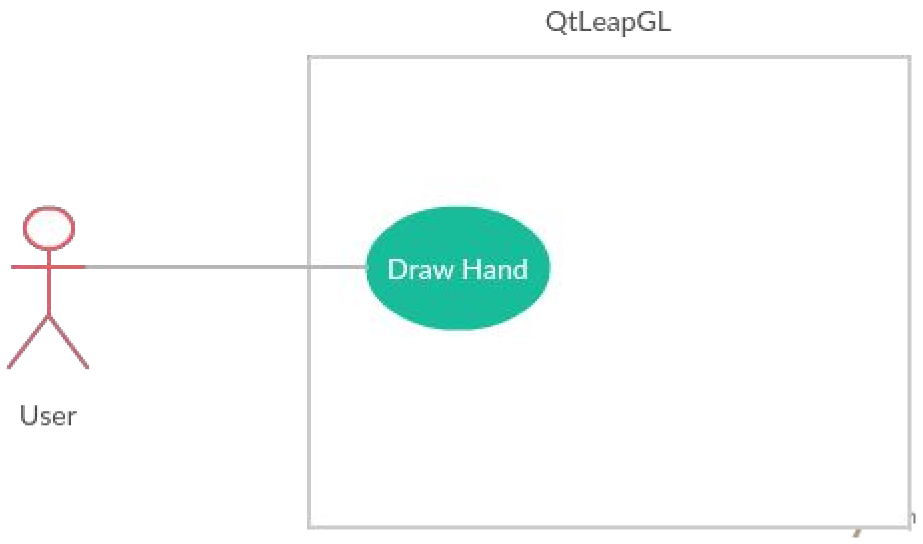

Figure 7 - LeapMotion Visualizer Use Case Diagram

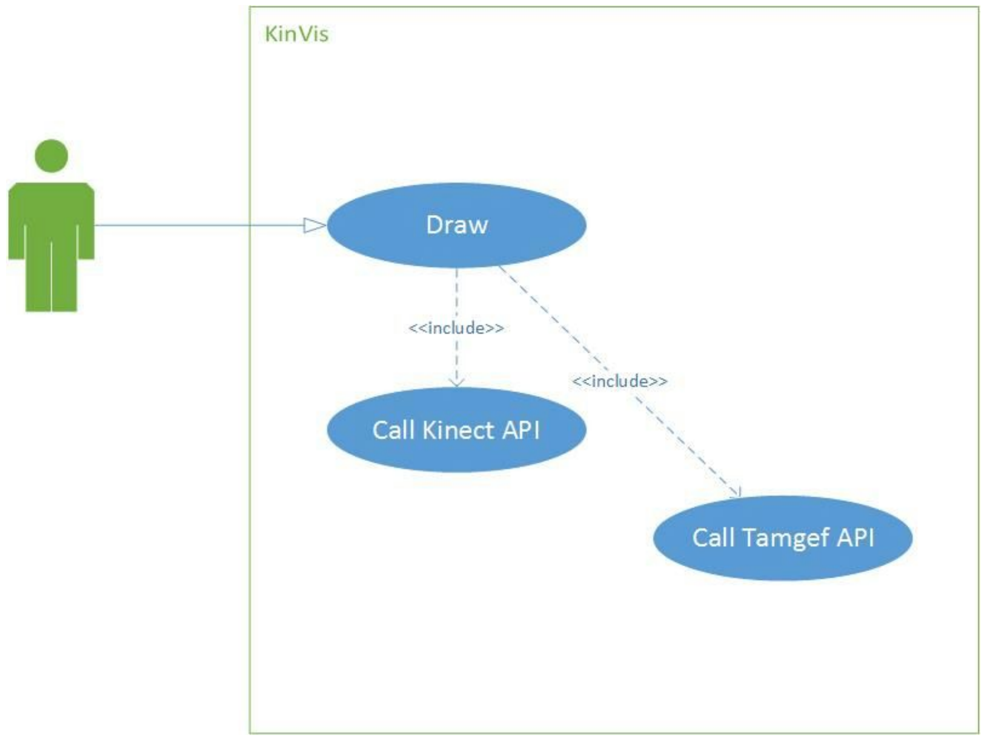

Figure 8 - Kinect Visualizer Use Case Diagram

Dynamic UML Diagrams

API Dynamic UML Diagrams

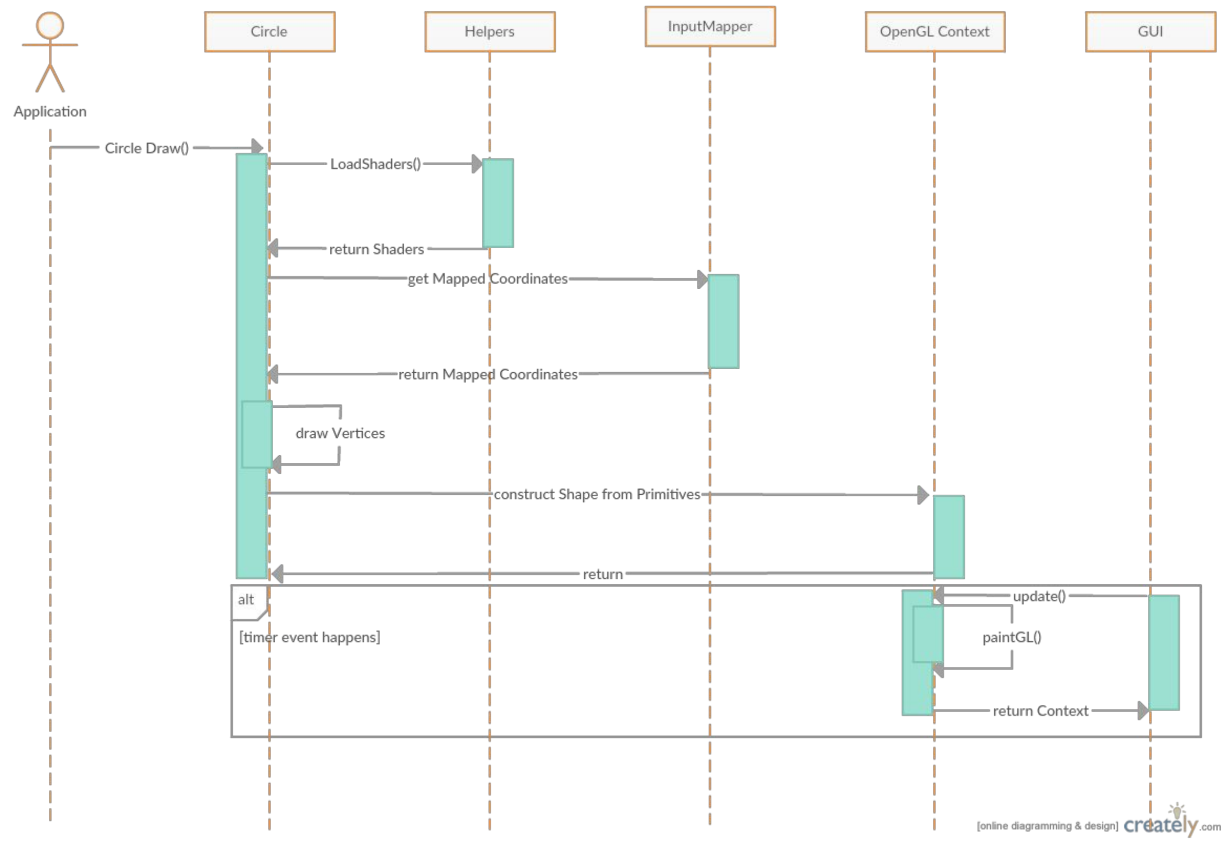

Figure 9 - Circle Draw Sequence Diagram

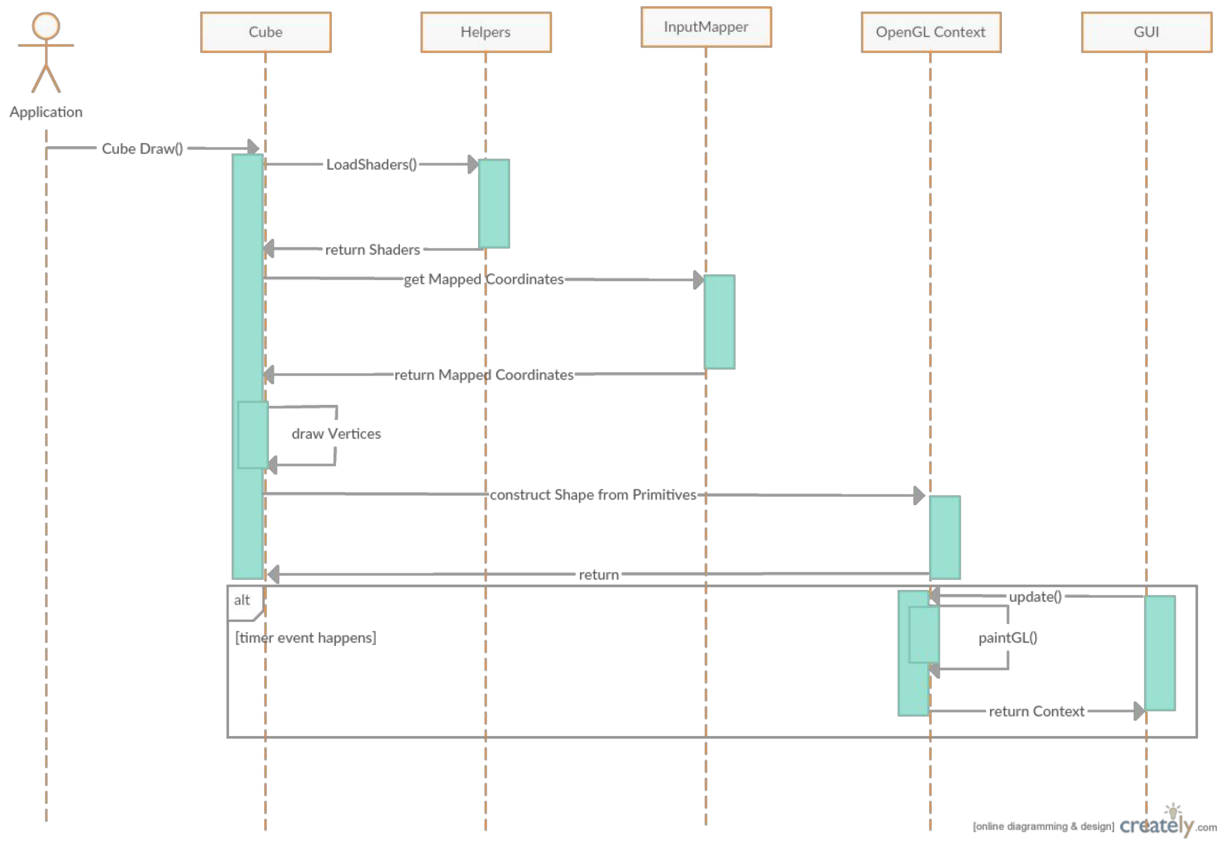

Figure 10 - Cube Draw Sequence Diagram

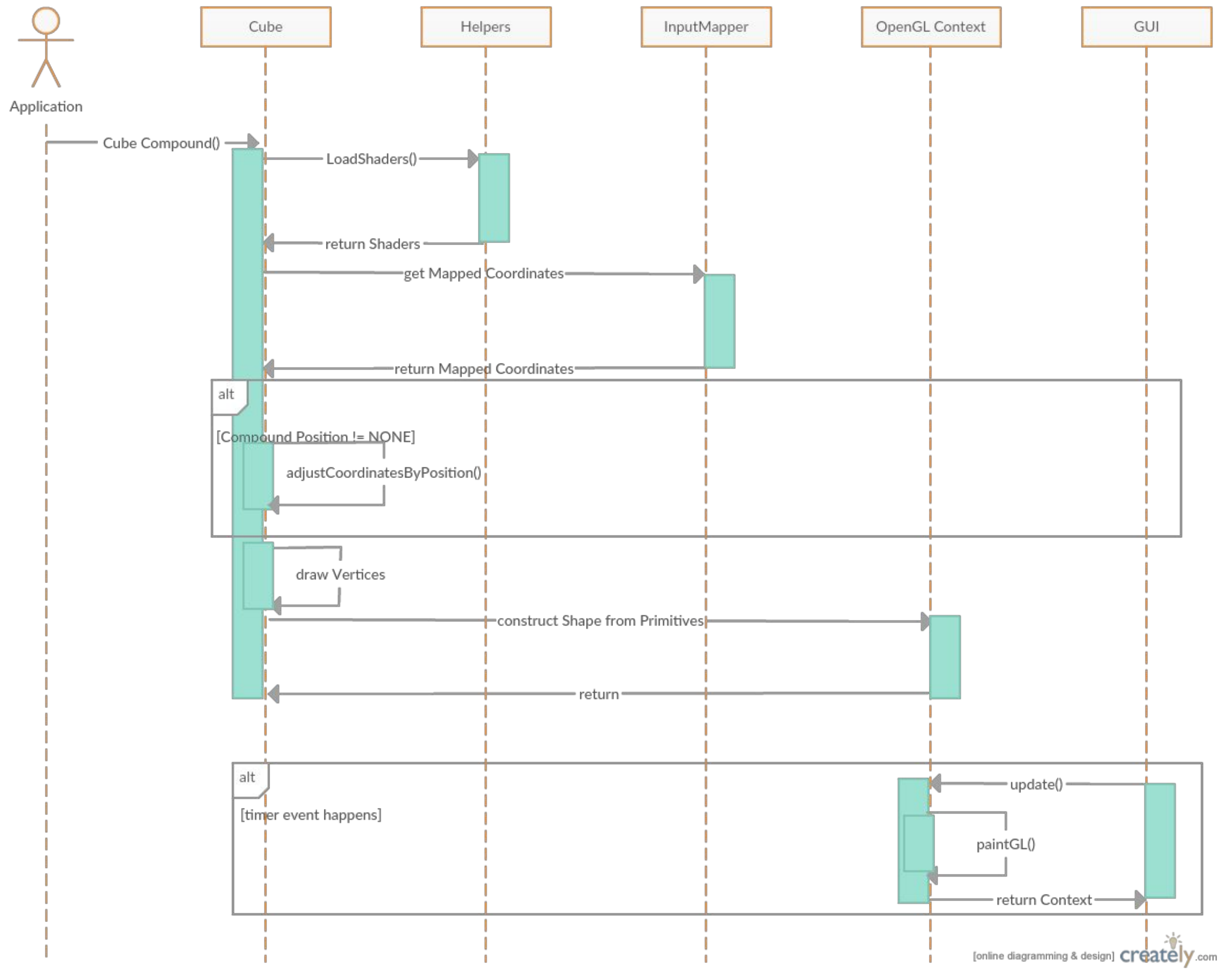

Figure 11 - Cube Compound Sequence Diagram

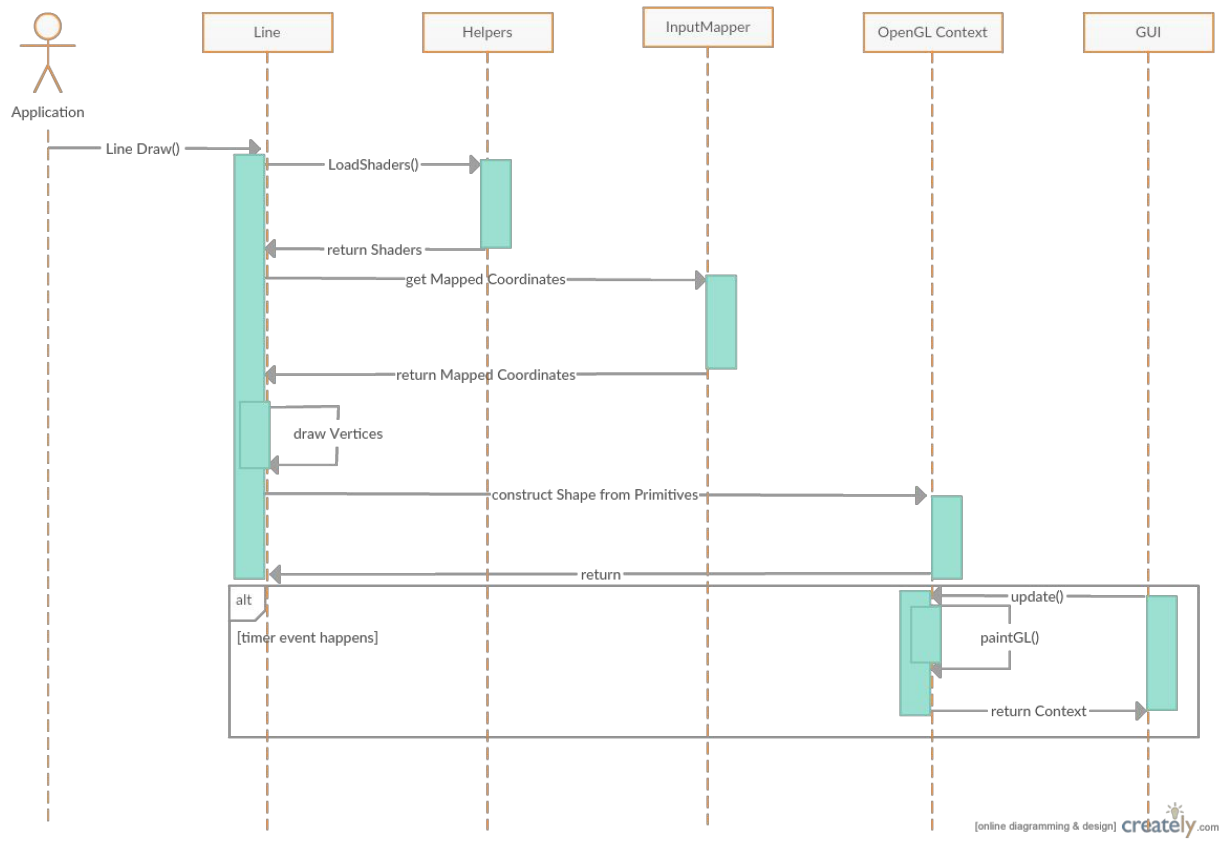

Figure 12 - Line Draw Sequence Diagram

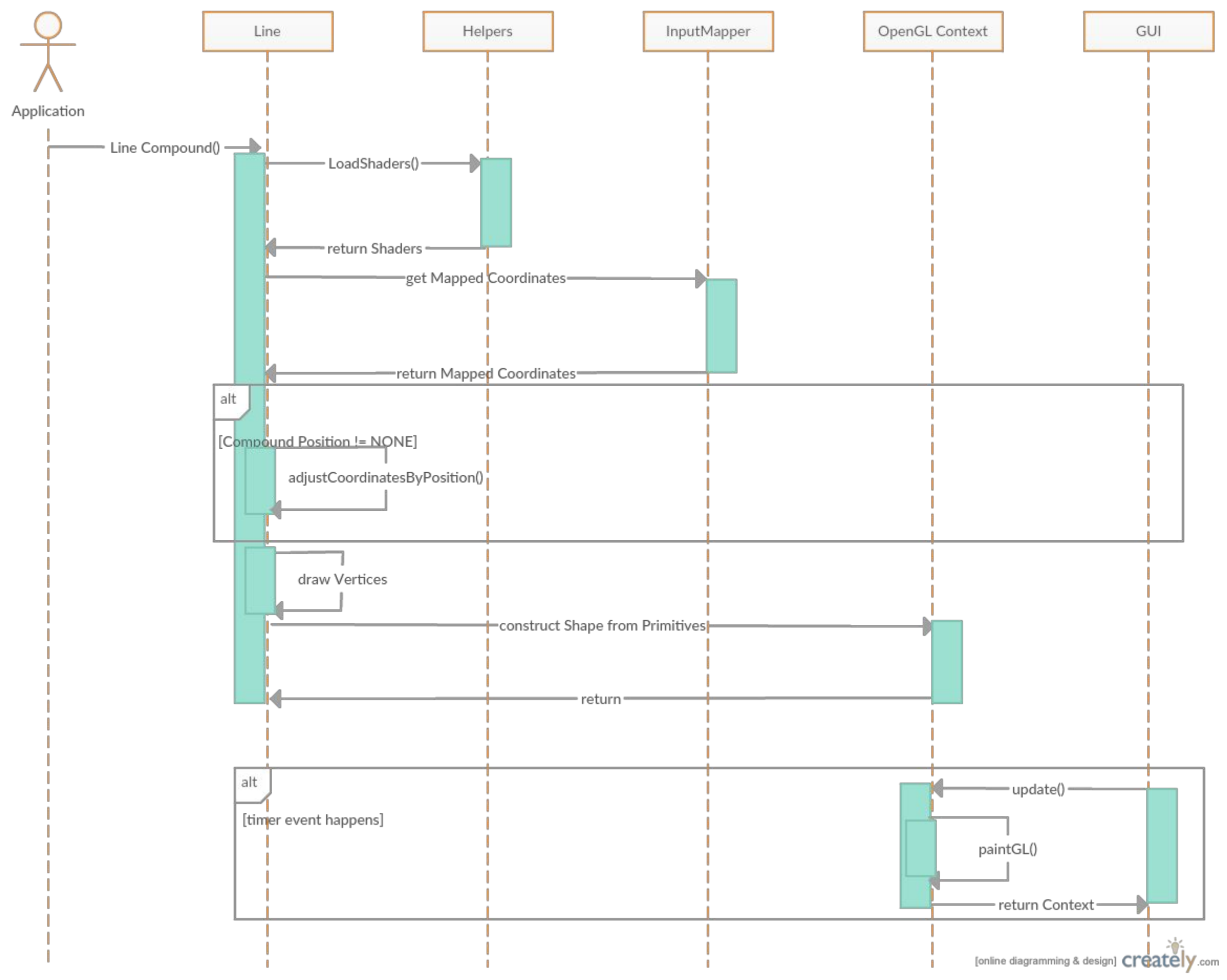

Figure 13 - Line Compound Sequence Diagram

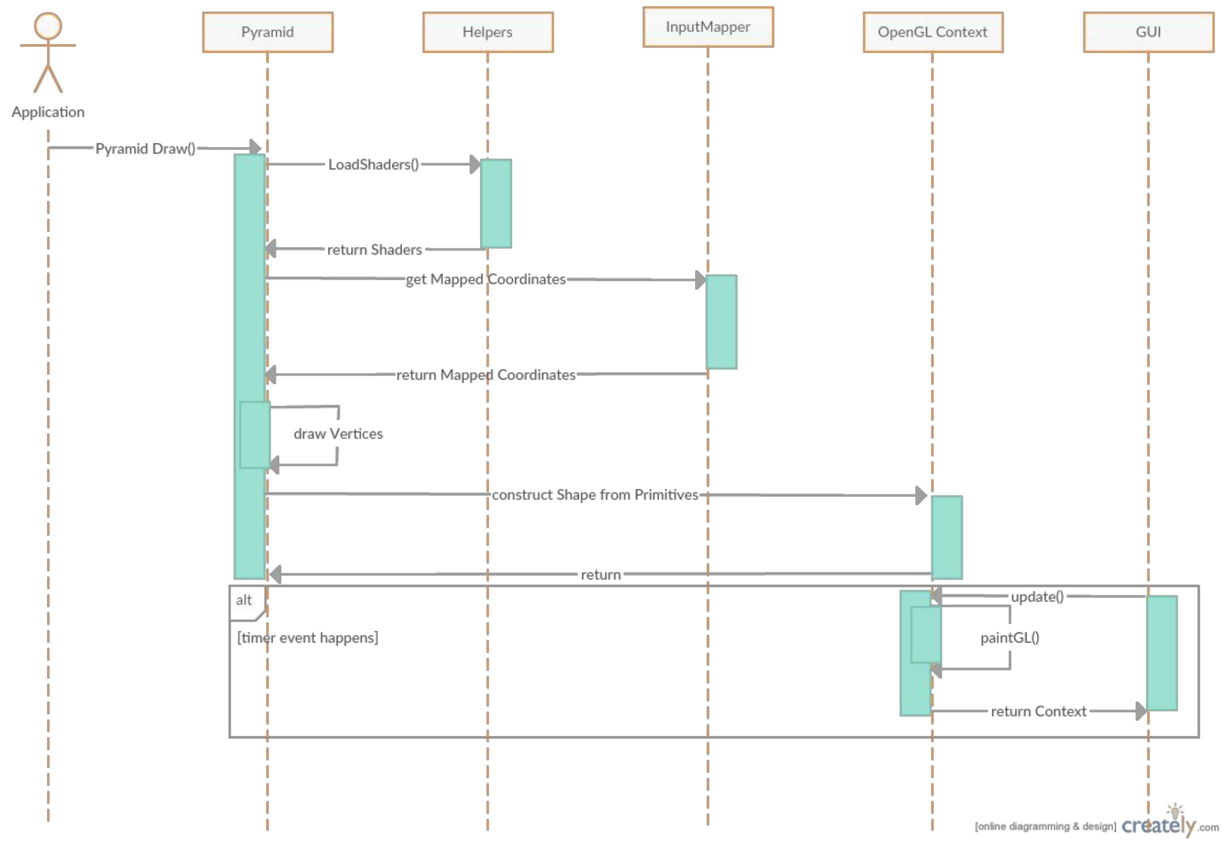

Figure 14 - Pyramid Draw Sequence Diagram

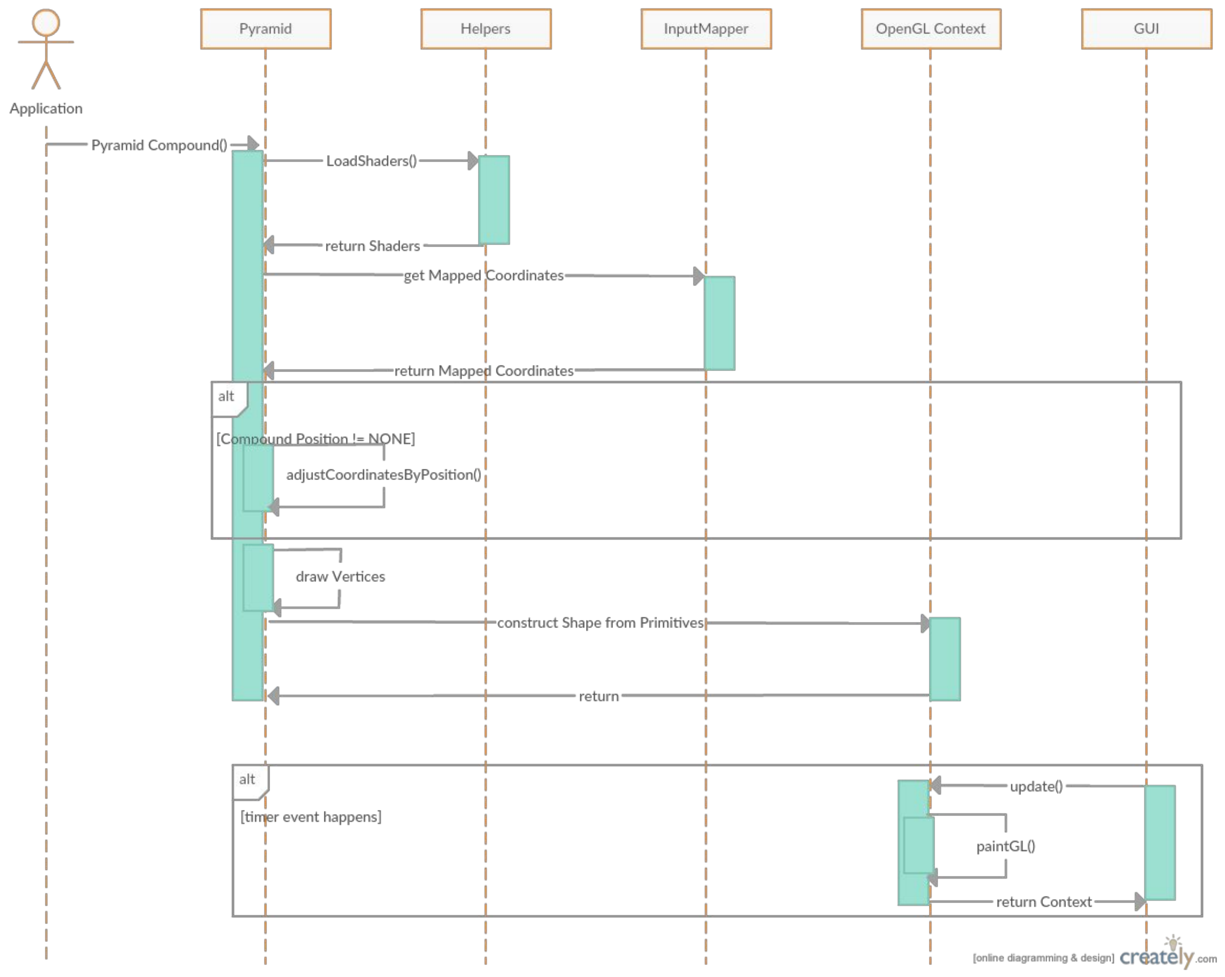

Figure 15 - Pyramid Compound Sequence Diagram

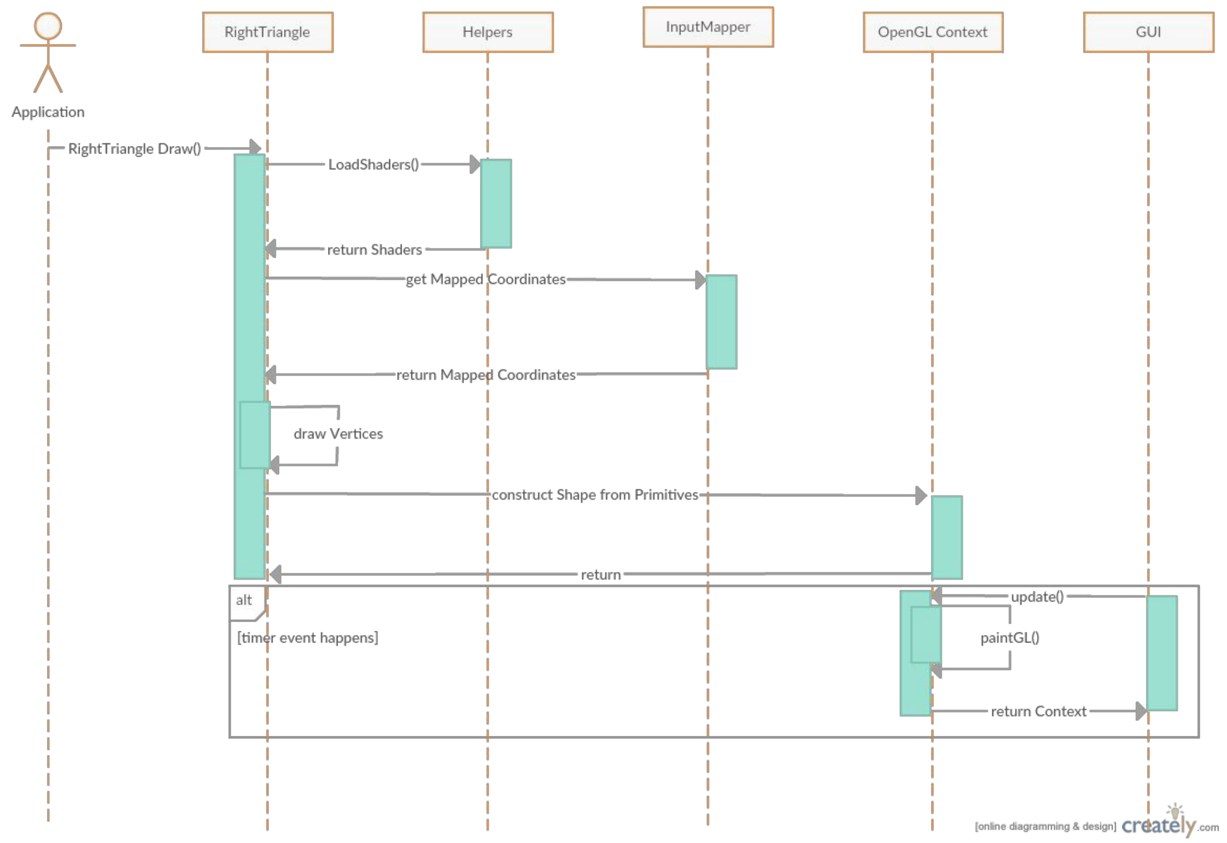

Figure 16 - Right Triangle Draw Sequence Diagram

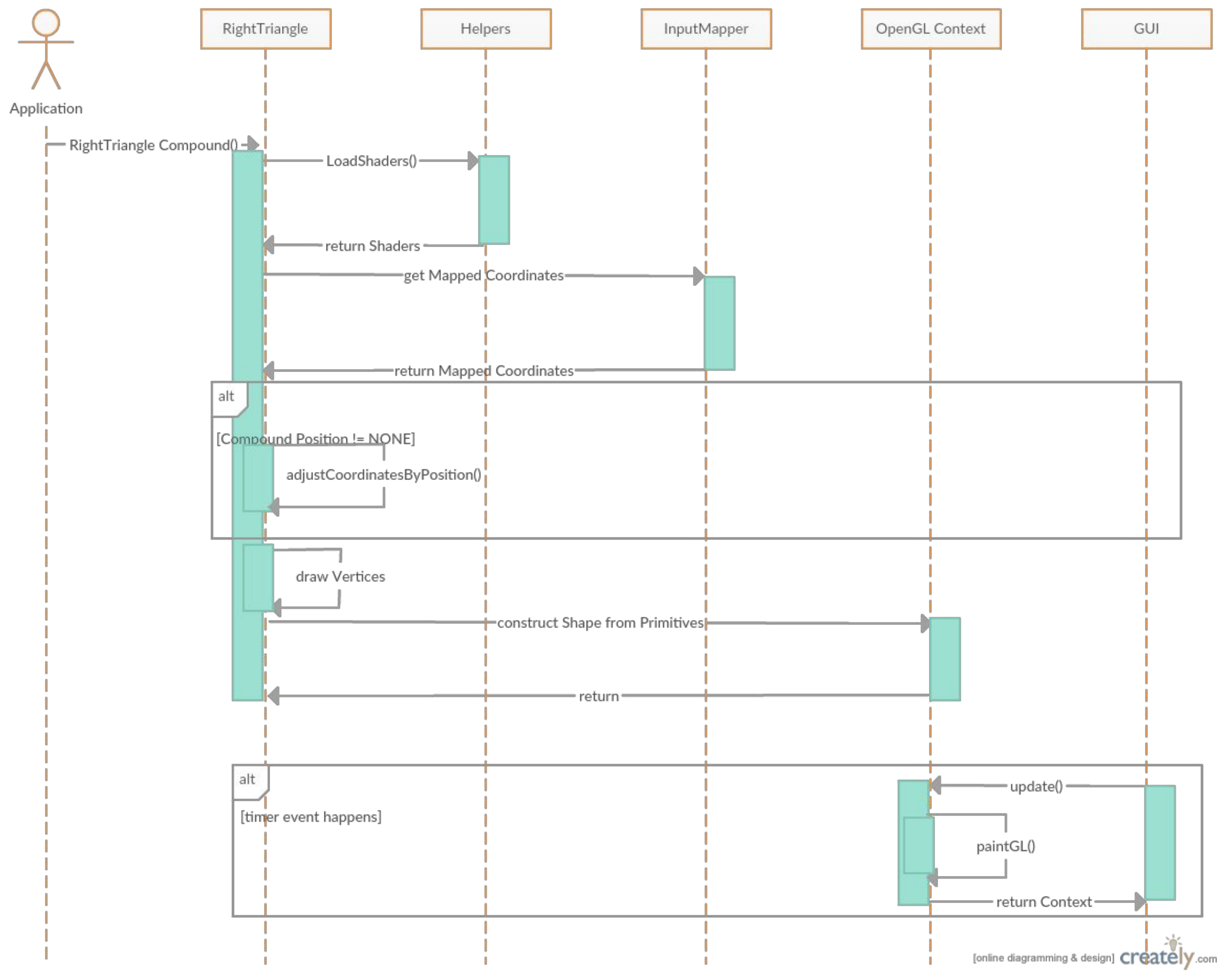

Figure 17 - Right Triangle Compound Sequence Diagram

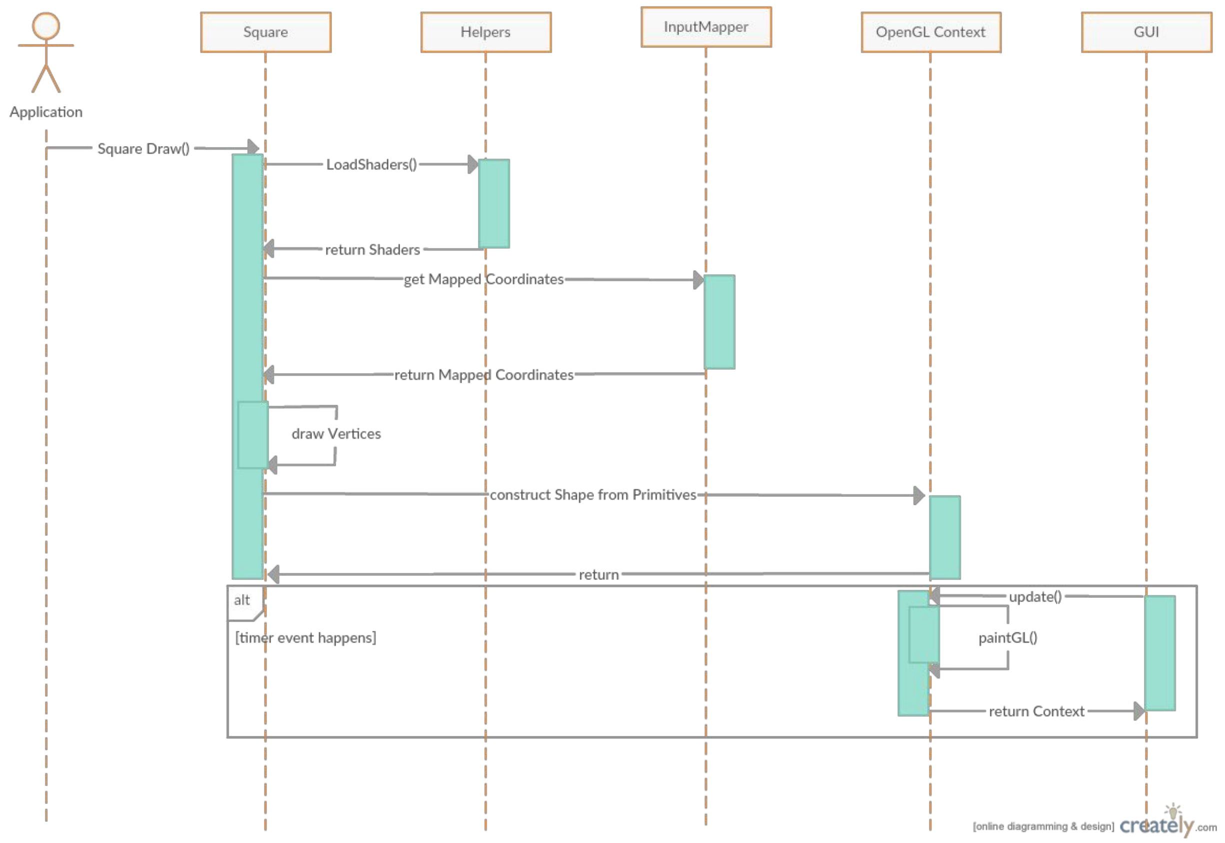

Figure 18 - Square Draw Sequence Diagram

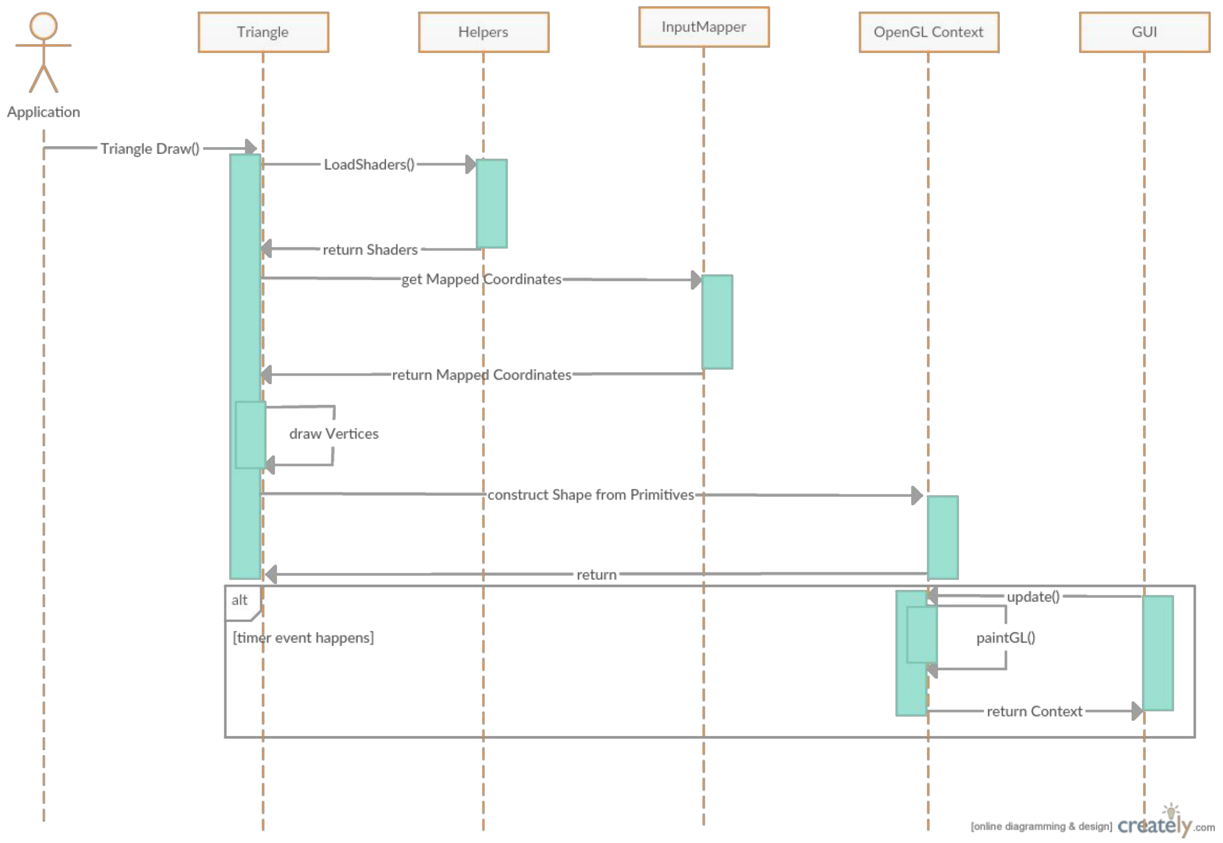

Figure 19 - Triangle Draw Sequence Diagram

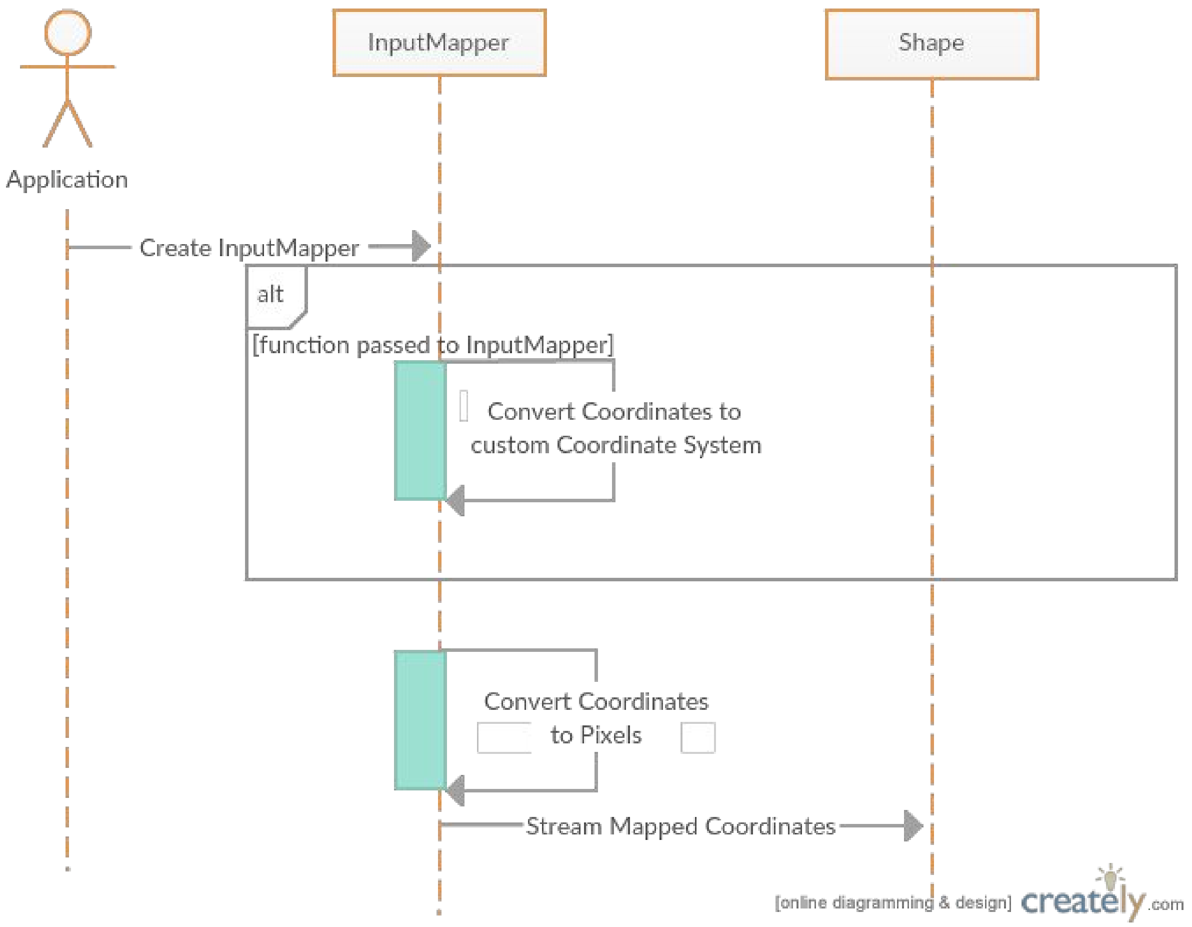
InputMapper
Shape
Application
Create InputMapper
alt
[function passed to InputMapper]
Convert Coordinates to custom Coordinate System
Convert Coordinates to Pixels
Stream Mapped Coordinates
[online diagramming & design] creately.com

Figure 20 - Input Mapping Sequence Diagram

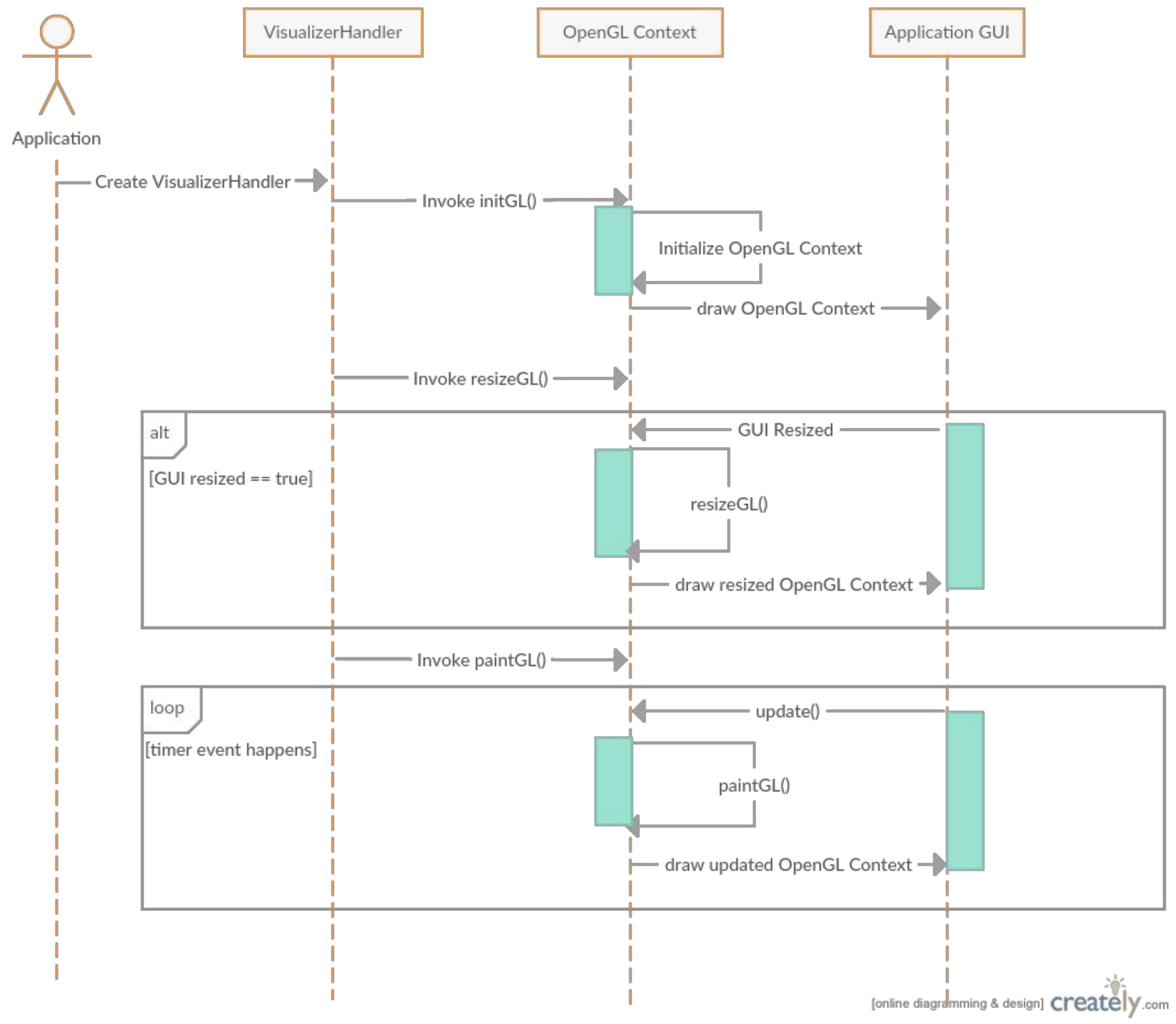

Figure 21 - OpenGL Context Creation Sequence Diagram

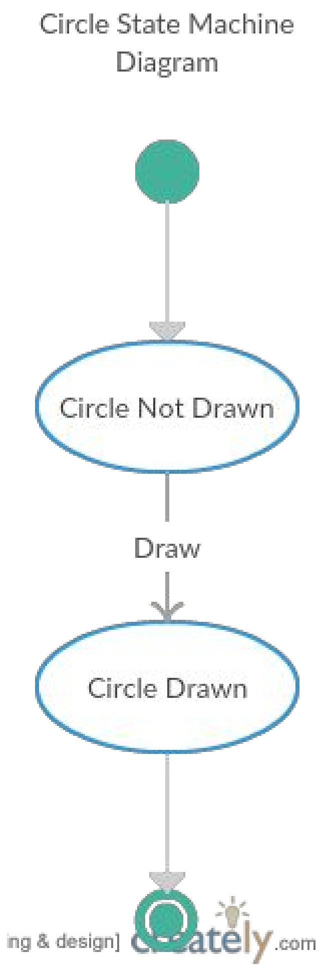

Figure 22 - Circle State Machine

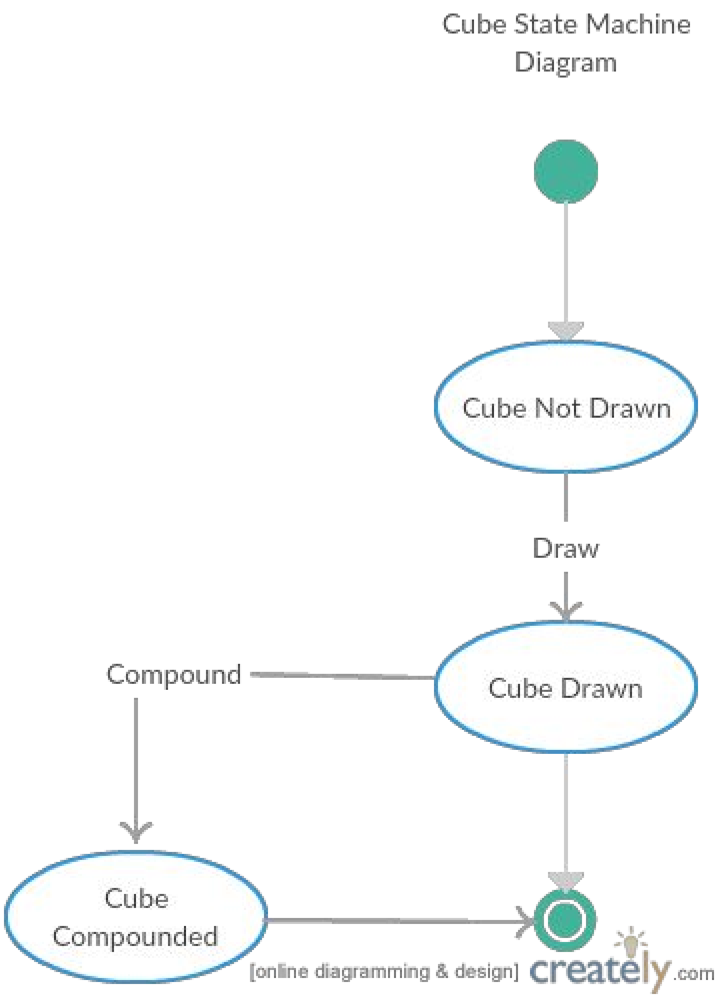
Cube State Machine Diagram
Cube Not Drawn
Draw
Cube Drawn
Compound
Cube Compounded
[online diagramming & design] creately.com

Figure 23 - Cube State Machine

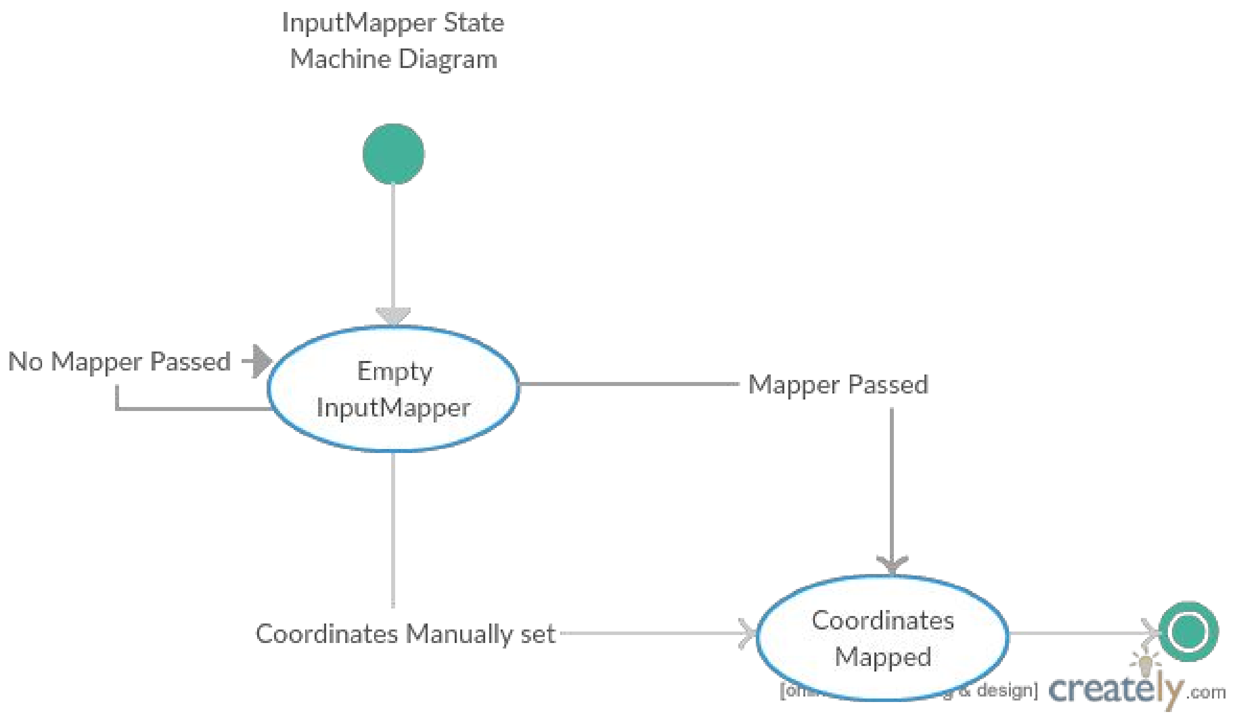

Figure 24 - InputMapper State Machine

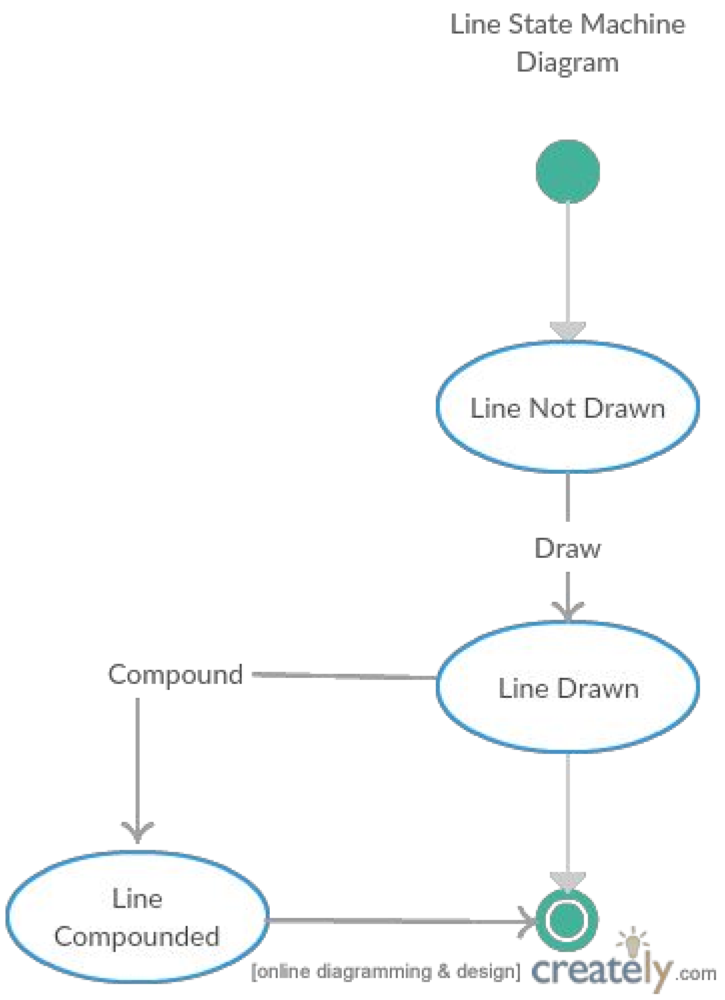
Line State Machine Diagram
Line Not Drawn
Draw
Line Drawn
Compound
Line Compounded
[online diagramming & design] creately.com

Figure 25 - Line State Machine

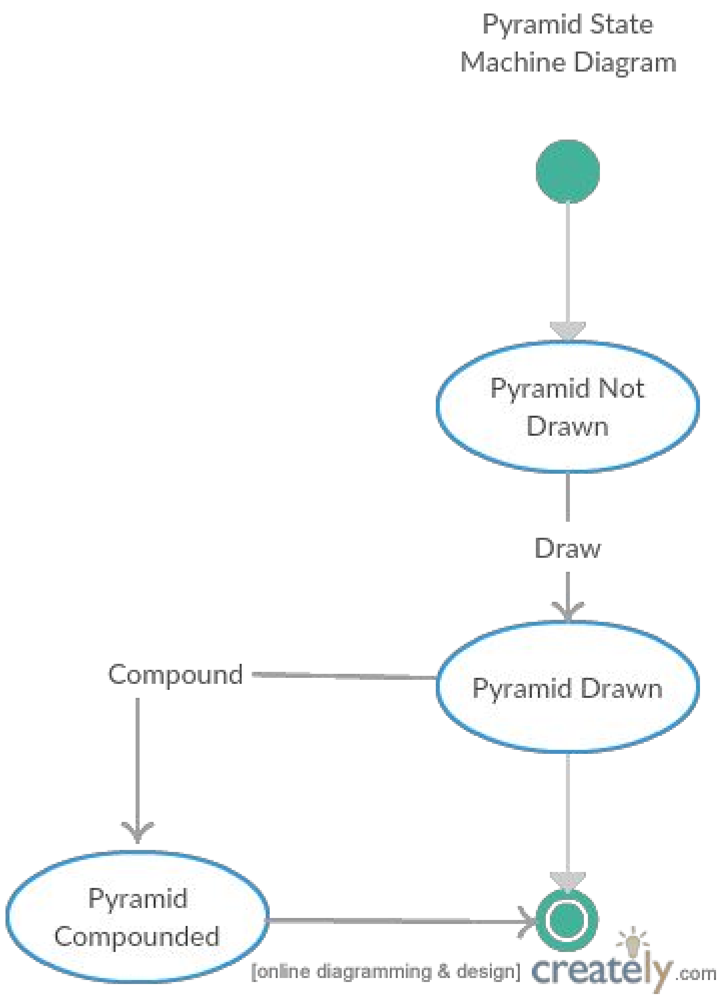

Figure 26 - Pyramid State Machine

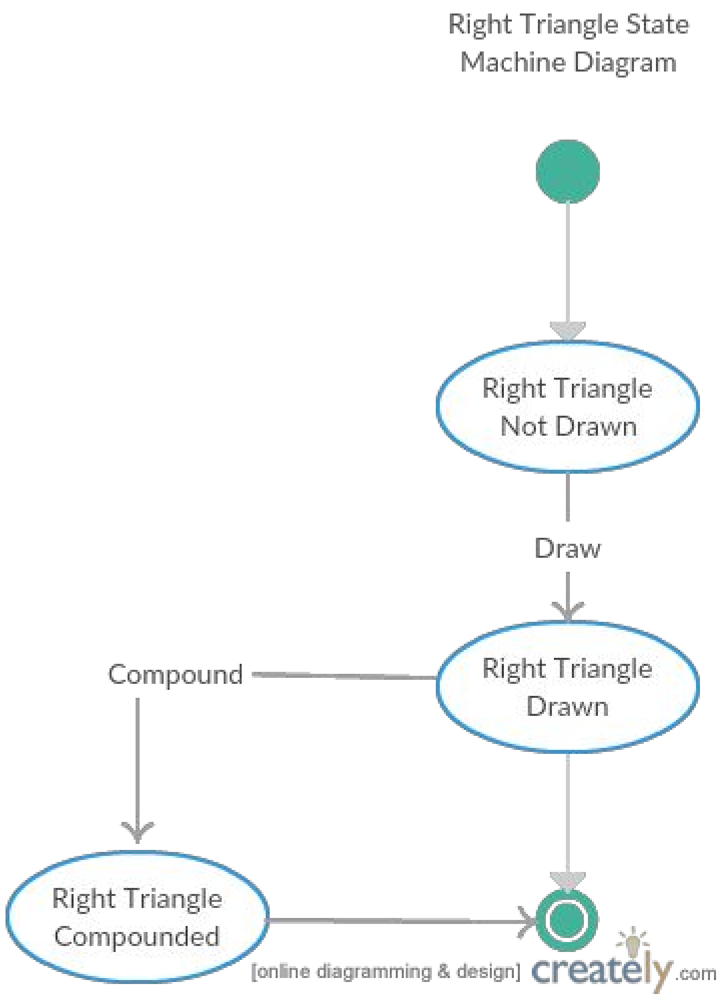
Right Triangle State Machine Diagram
Right Triangle Not Drawn
Draw
Right Triangle Drawn
Compound
Right Triangle Compounded
[online diagramming & design] creately.com

Figure 27 - Right Triangle State Machine

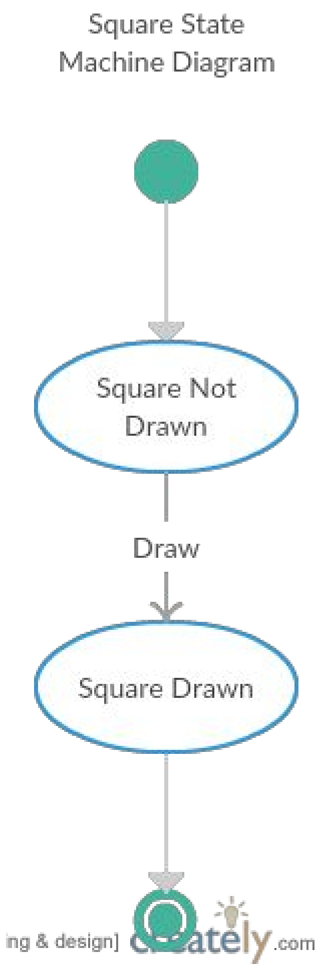

Figure 28 - Square State Machine

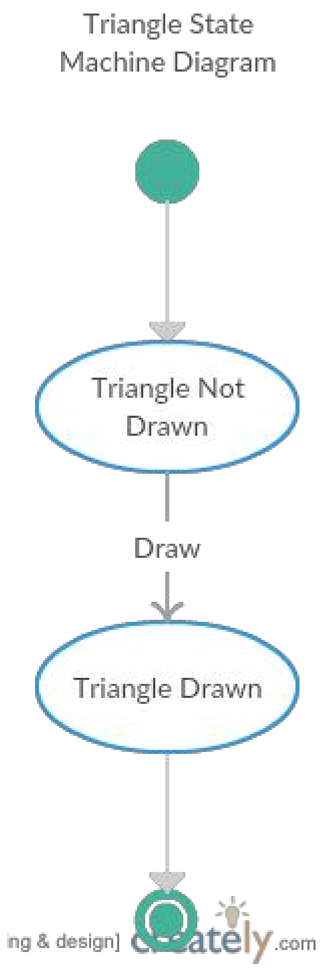

Figure 29 - Triangle State Machine

LeapMotion Visualizer Dynamic UML Diagrams

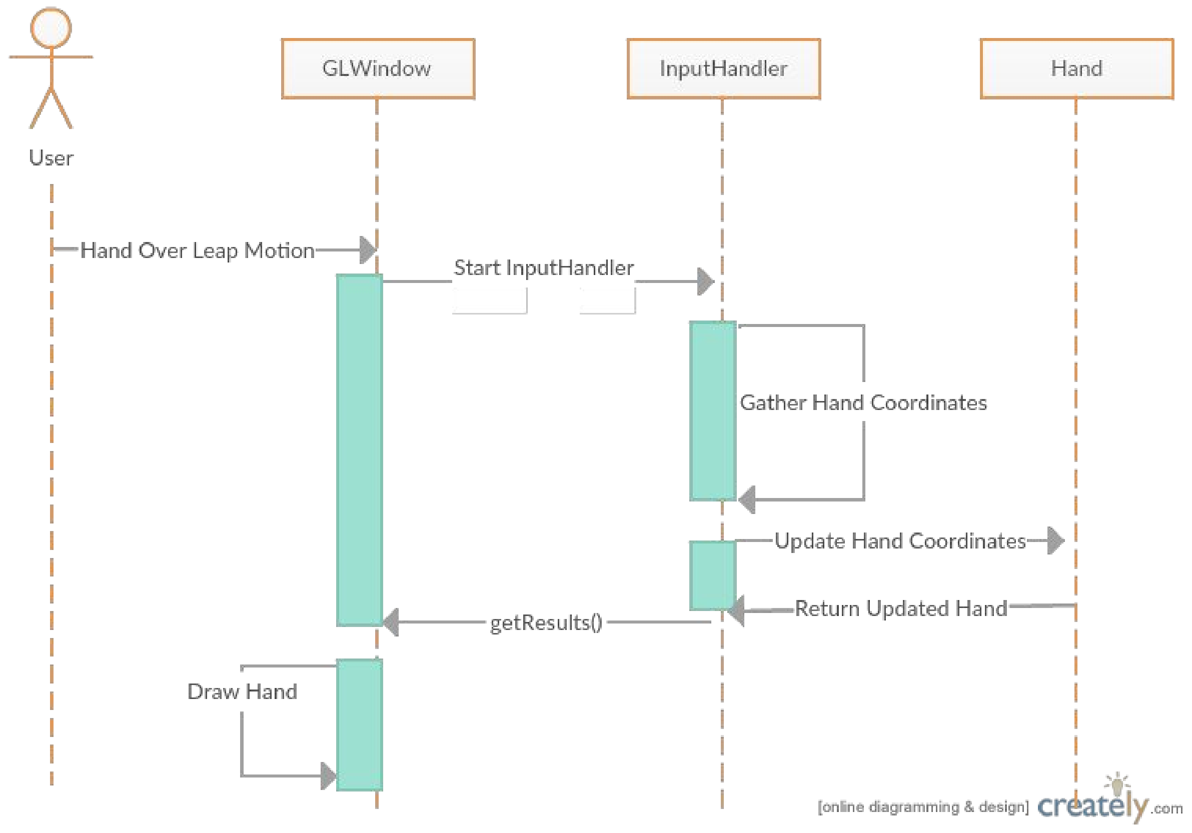

Figure 30 - Draw Hand Sequence Diagram

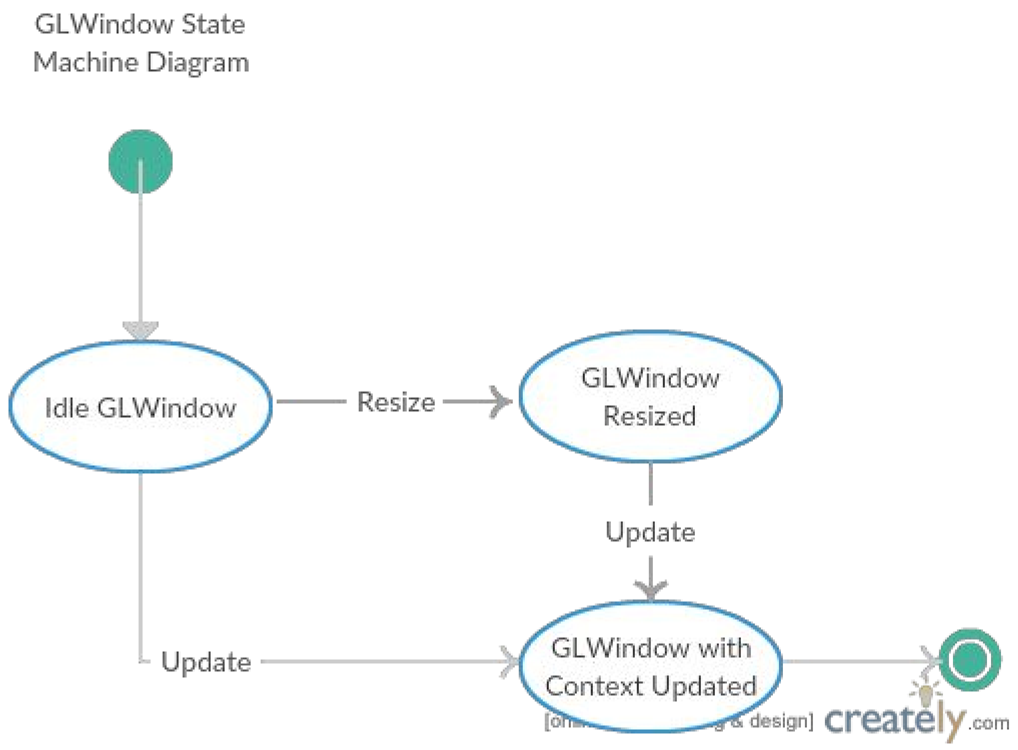

Figure 31 - GLWindow State Machine

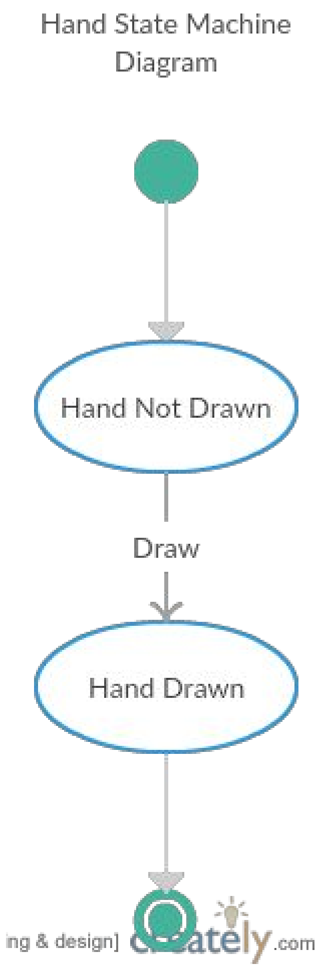

Figure 32 - Hand State Machine

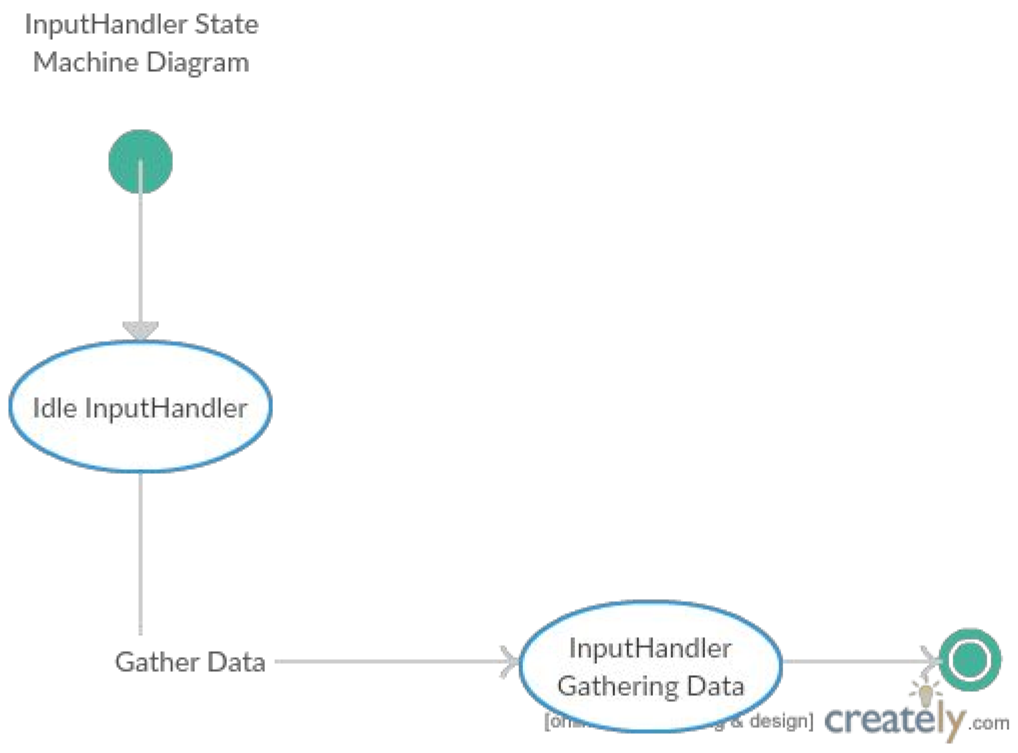
InputHandler State Machine Diagram
Idle InputHandler
Gather Data
InputHandler Gathering Data

Figure 33 - InputHandler State Machine

Kinect Visualizer Dynamic UML Diagrams

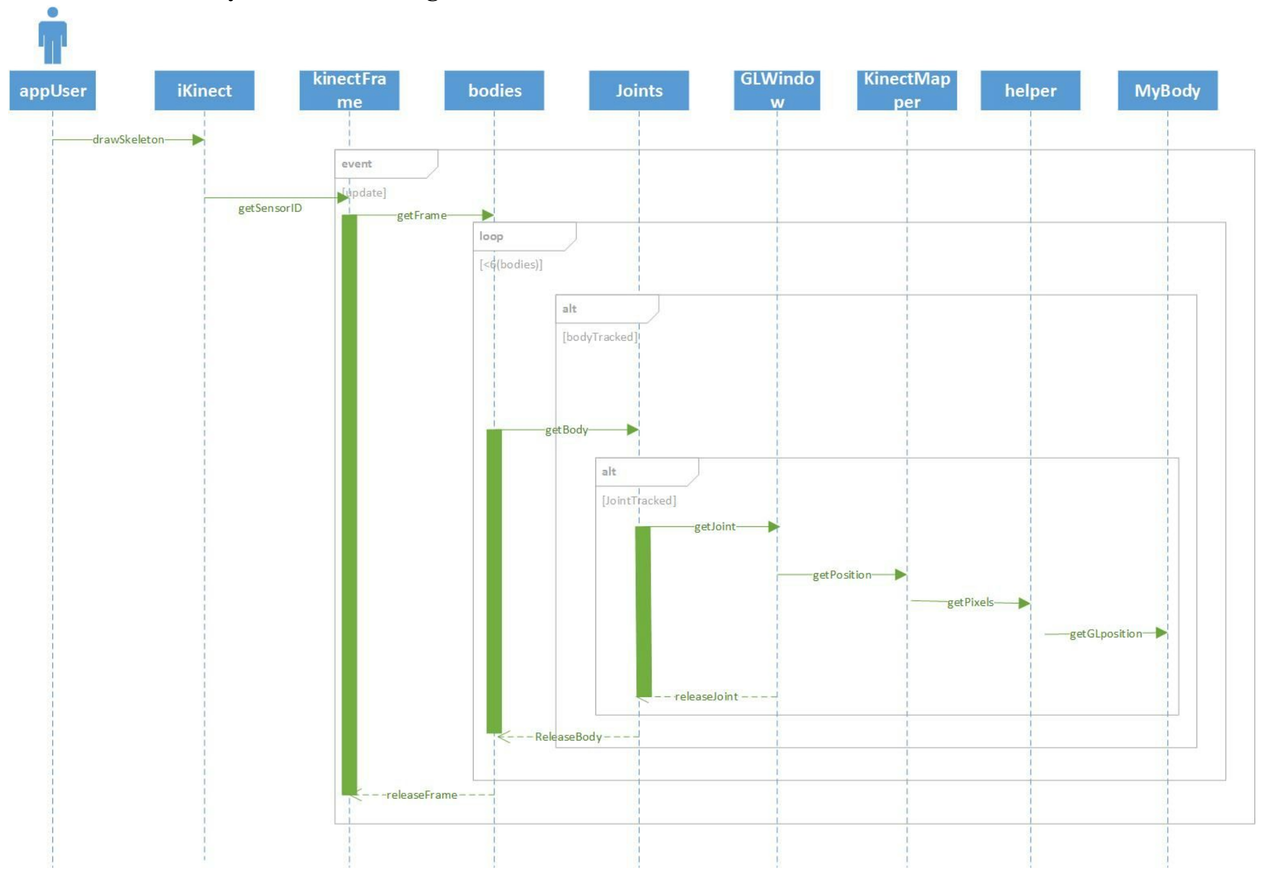

Figure 34 - draw skeleton Sequence Diagram

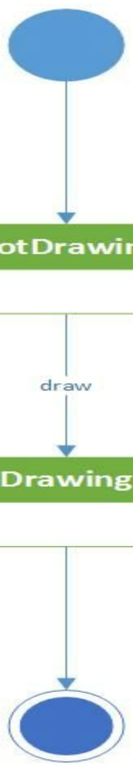

Figure 35 - draw state machine

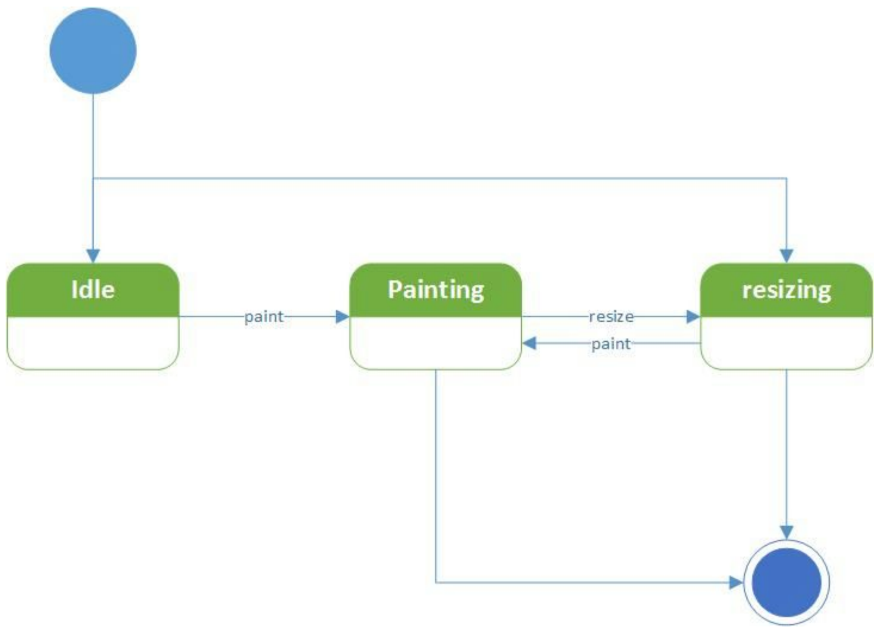

Figure 36 - GLWindow state machine

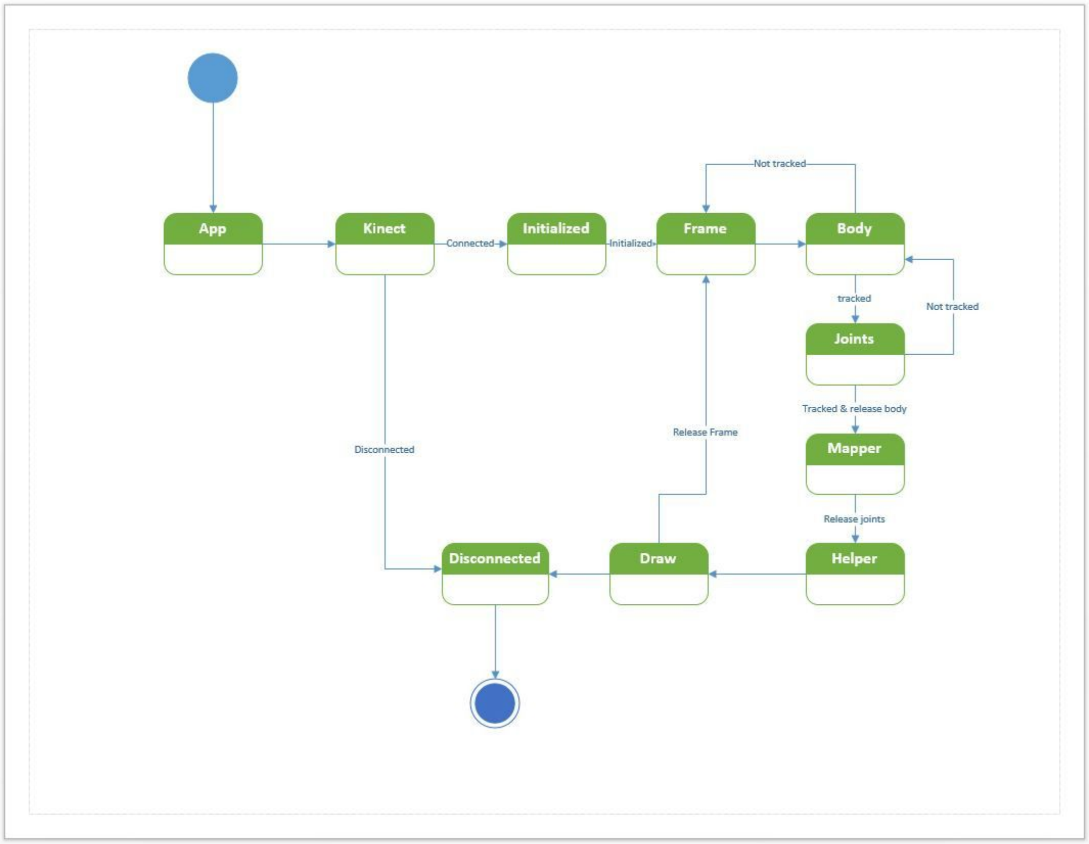

Figure 37 - Kinect Visualizer Activity Diagram

Appendix B - User Interface Design

The Previous TAMVisualizer works as demo for the TAMGeF API along with the LeapMotion and Kinect visualizers

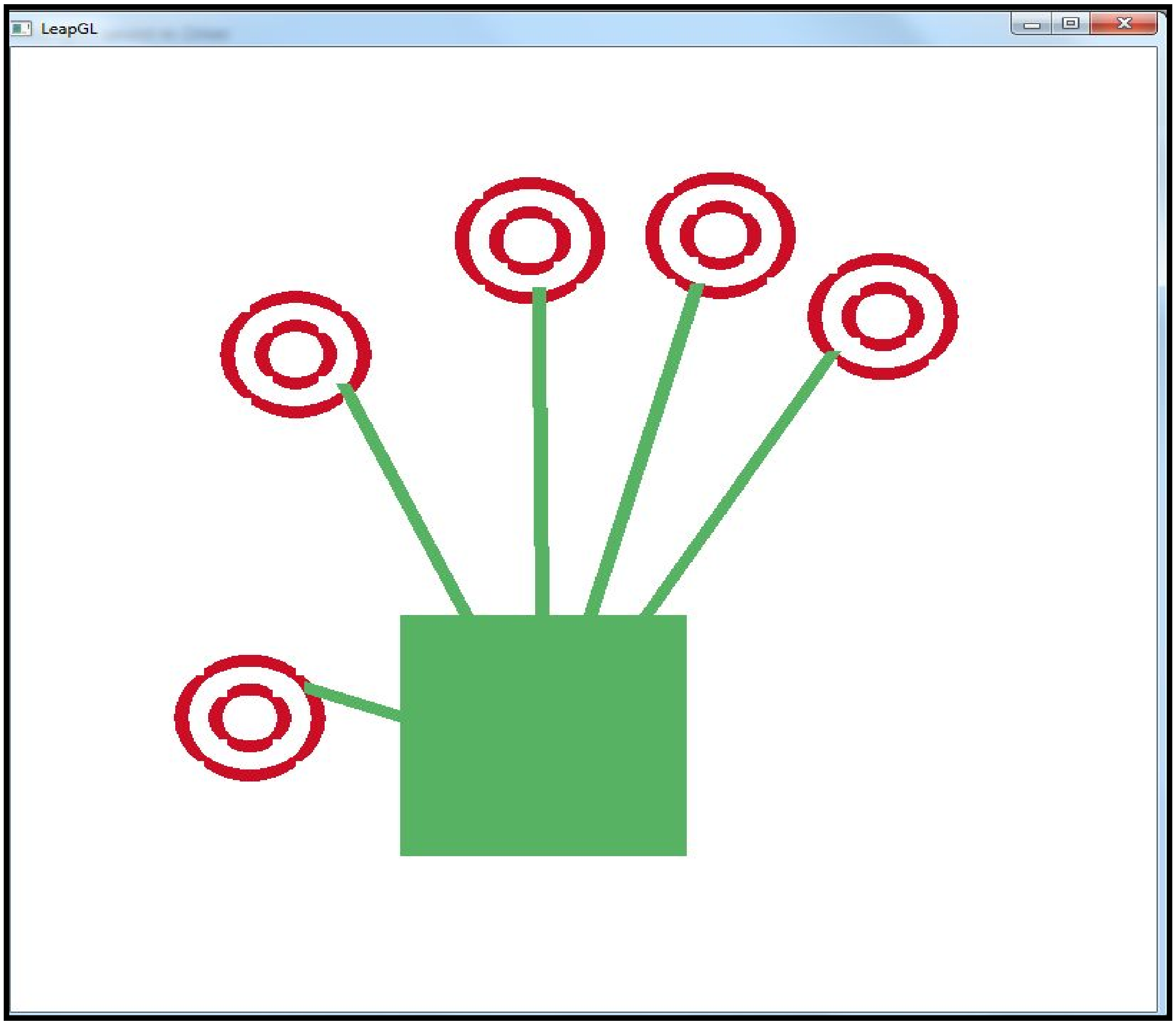

Figure 38 - LeapMotion Visualizer

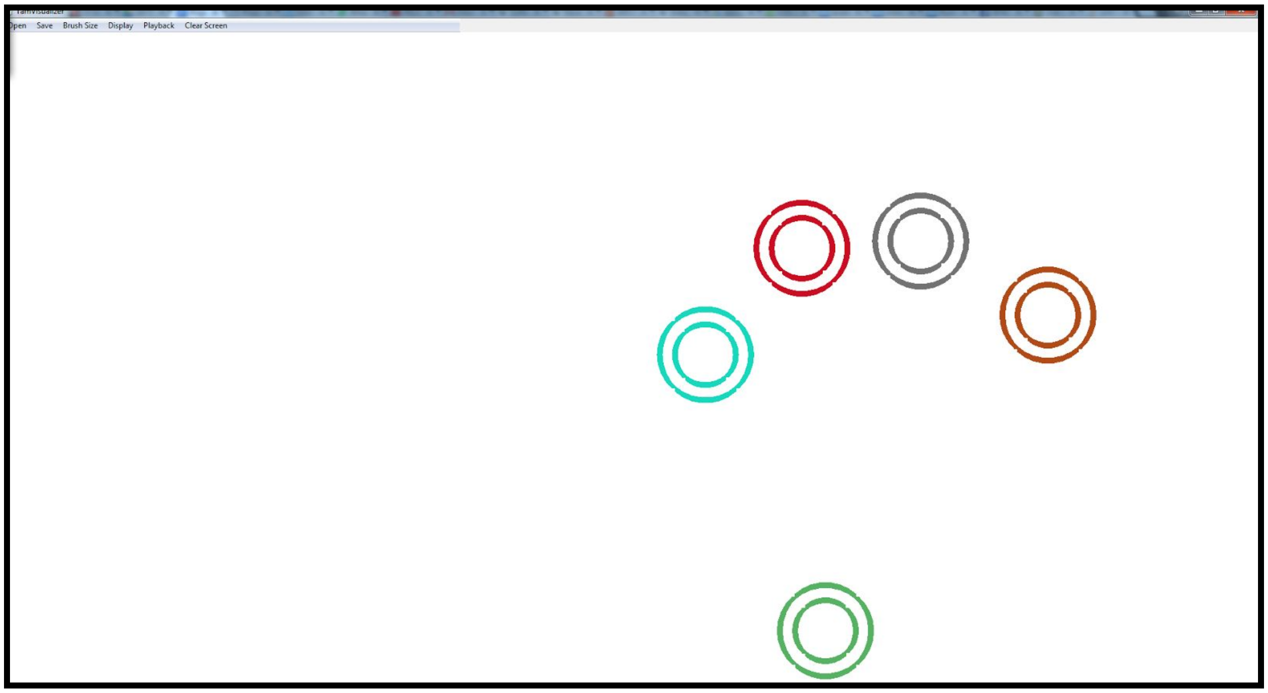

Figure 39 - TAMGeF API Circles(Each circle is composed of an outer and inner Circle)

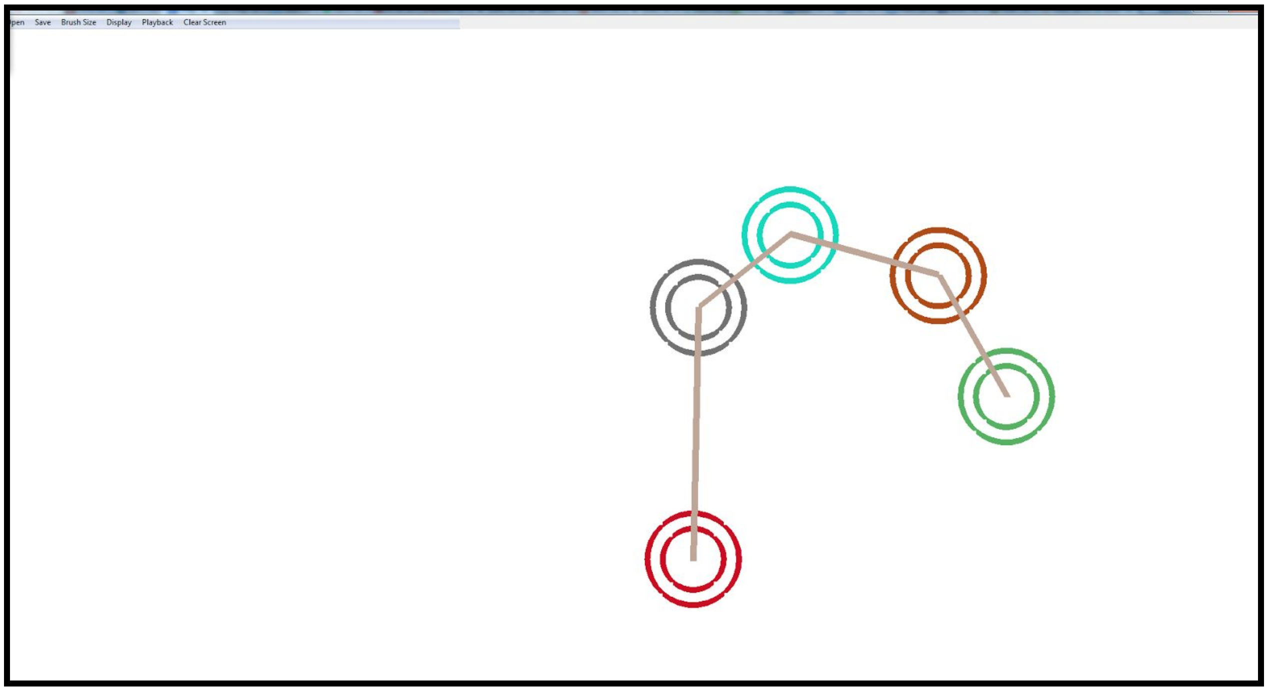

Figure 40 - TAMGeF API Displaying Line Connection

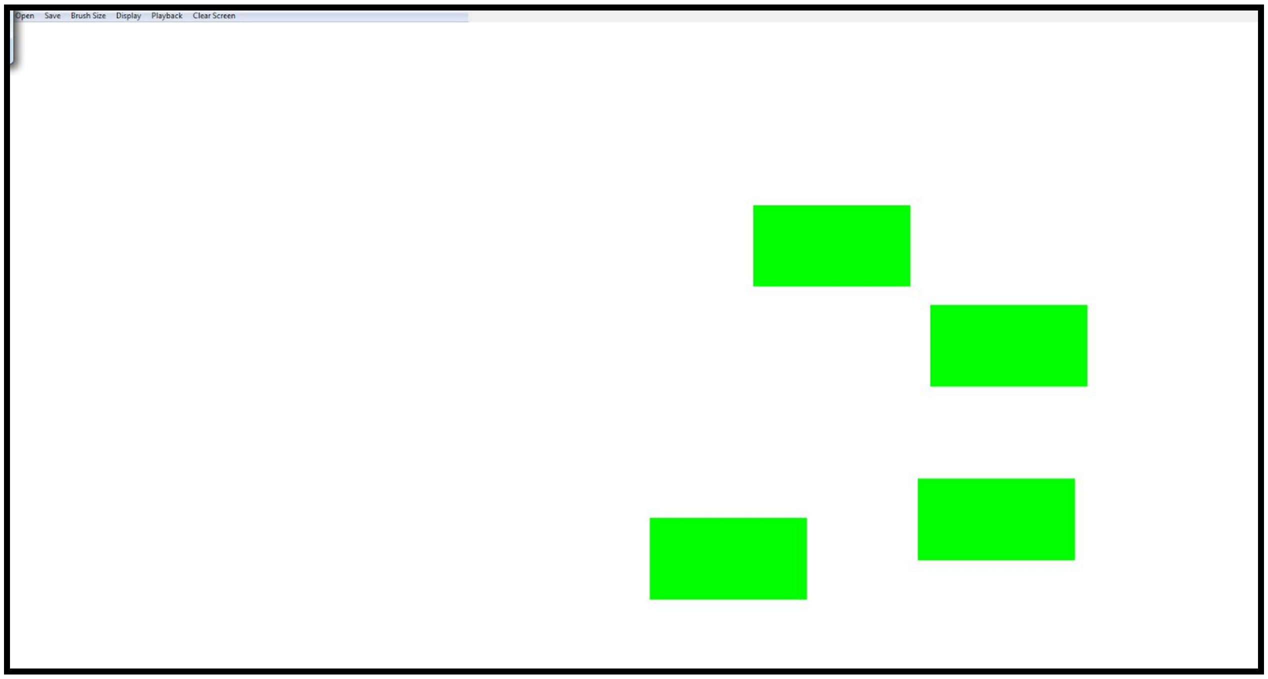

Figure 41 - TAMGeF API Displaying Rectangles achieved by compounding

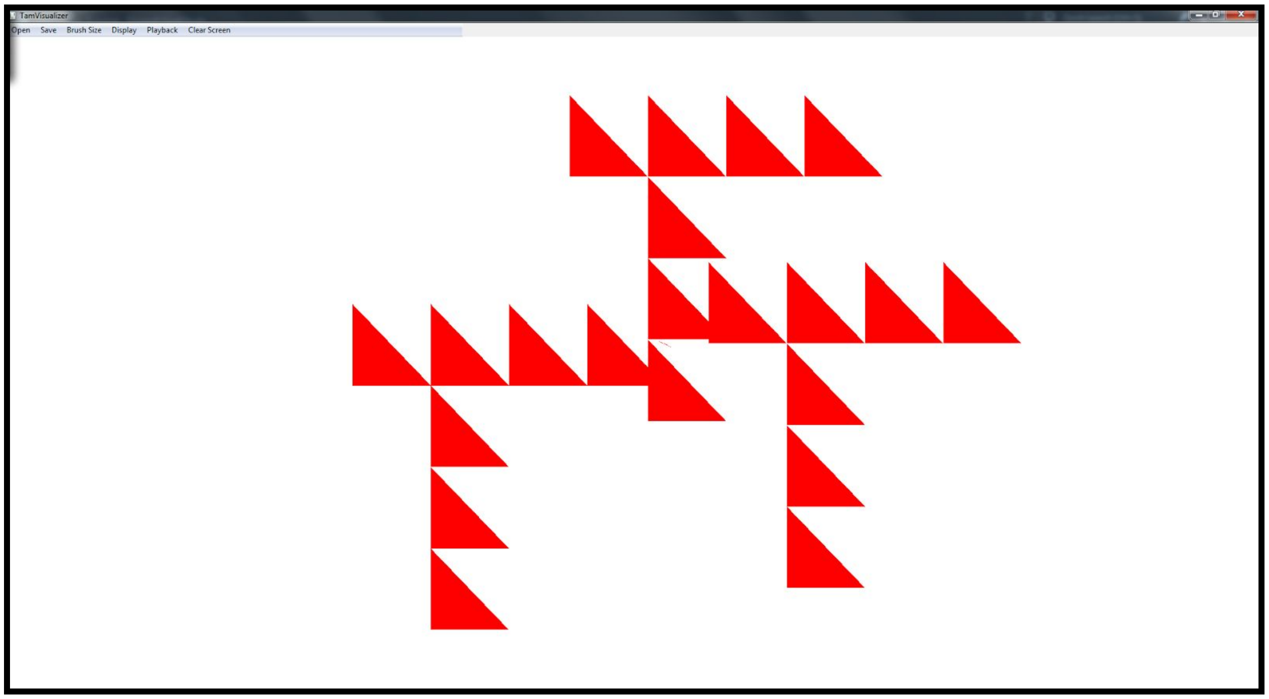

Figure 42 - TAMGeF API Displaying Compounded Right Triangles

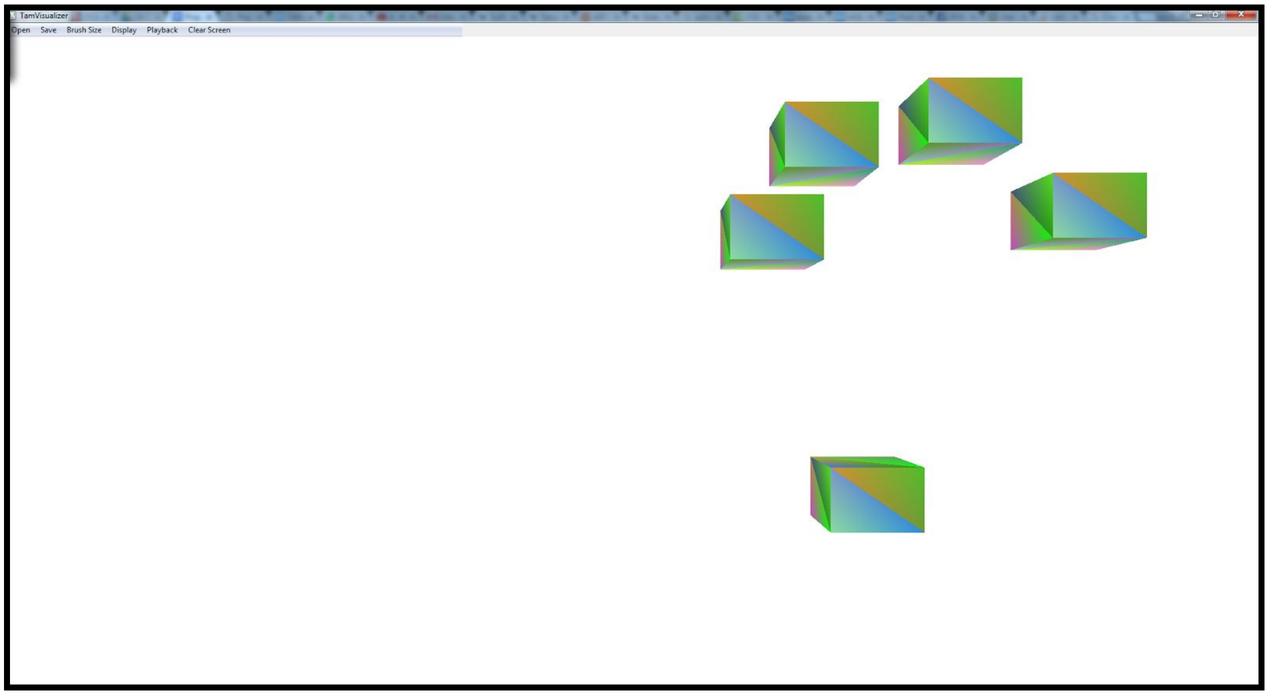

Figure 43 - TAMGeF API Displaying Cubes which vertices have each a different color

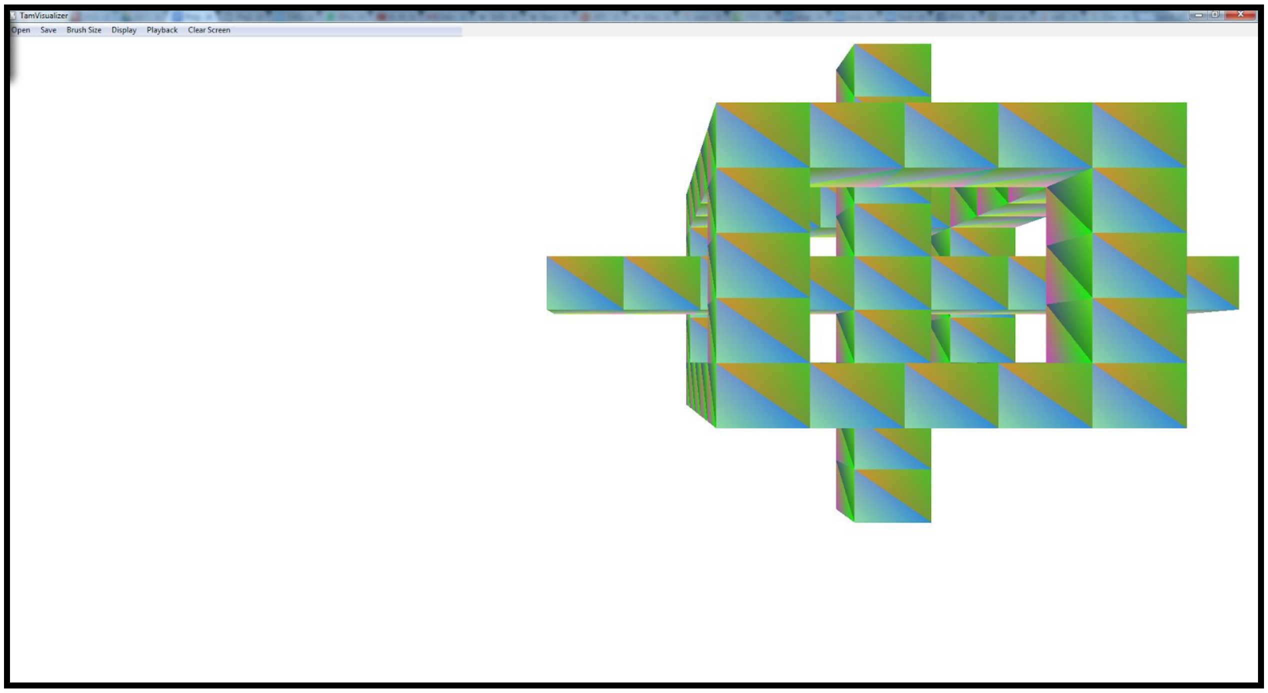

Figure 44 - TAMGeF API Displaying a structure achieved by compounding cubes

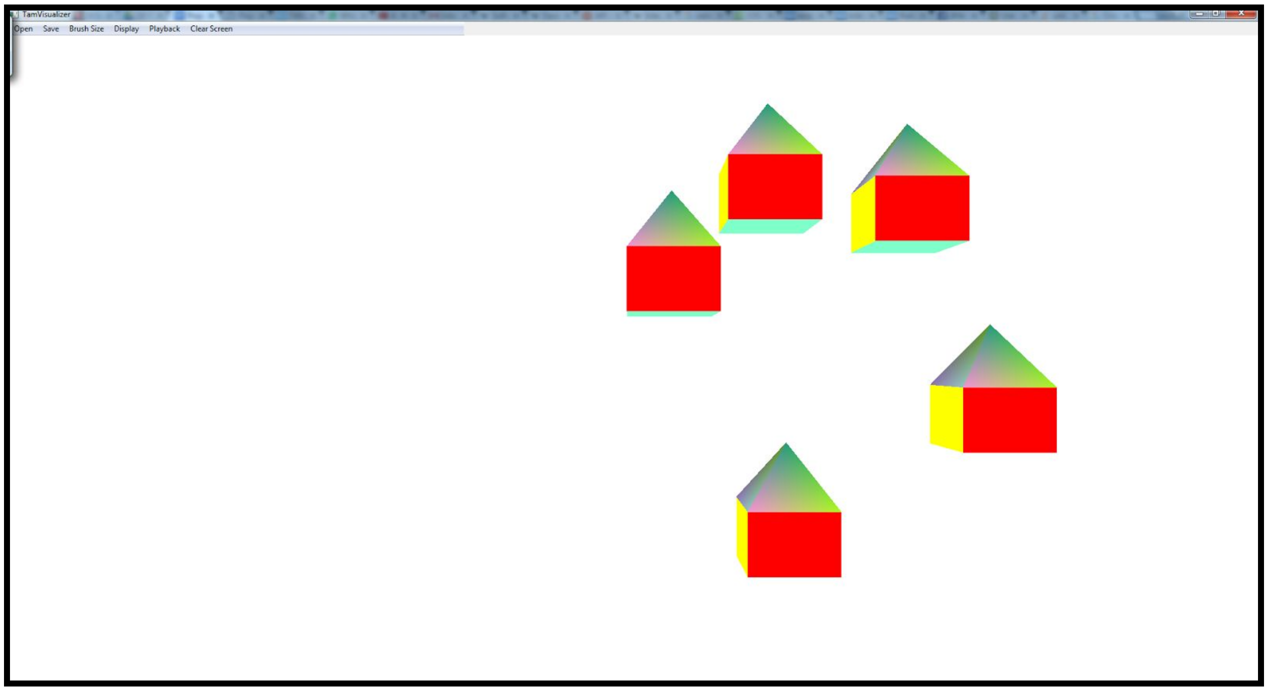

Figure 45 - TAMGeF API Displaying a Cube with custom color selection and a Pyramid compounded to it

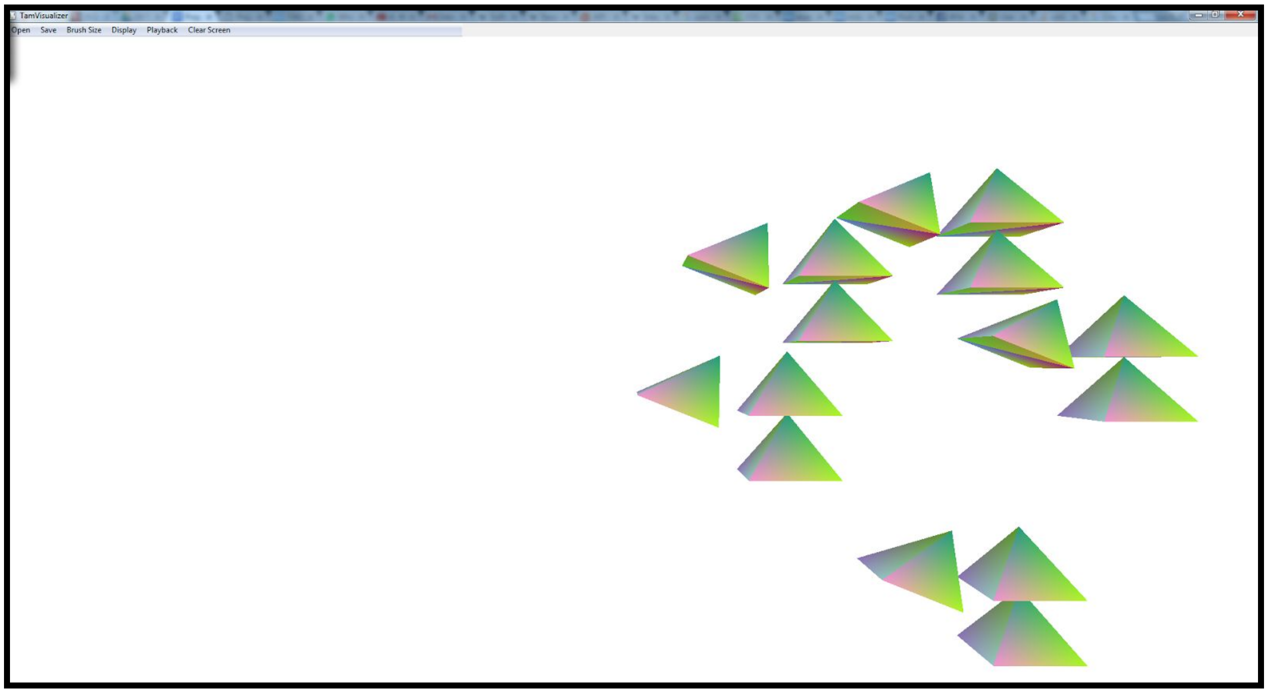

Figure 46 - TAMGeF API Displaying Pyramid compounding and rotation

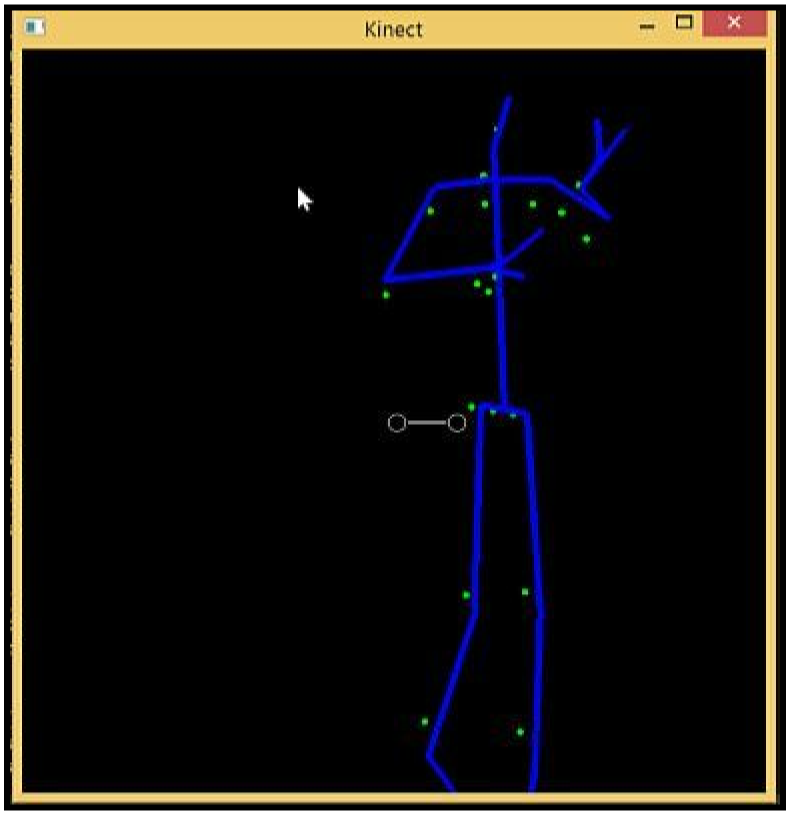

Figure 47 - Kinect Visualizer displaying a Skeleton drawn using the API

Appendix C - Sprint Review Reports

Sprint 1 Report

Date: September 11, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Fixed a drawing bug with the previous TAMVisualizer(Alberto)
    - Setup Development Environment for the TAMVisualizer and API(Alberto)
    - Researched API Design(Alberto)
    - Setup Development Environment for LeapMotion(David)
    - Studied C++(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for meetings for discussion of the API and OpenGL
    - Added user stories which involved reading to expand the team members knowledge
    - Added user stories for the development of the LeapMotion data application

Sprint 2 Report

Date: September 25, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Migrated the previous Visualizer to OpenGL Programmable Pipeline(Alberto)
    - Designed the Visualizing API by using Diagrams(Alberto)
    - Created LeapMotion data Application(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for the actual implementation of the visualizing API
    - Added user stories for the creation of design documentation of the API
    - Added user stories for creation of the LeapMotion application for windows

Sprint 3 Report

Date: October 9, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Started implementation of the TAMGeF API(Alberto)
    - Added automatic OpenGL Context creation to the API(Alberto)
    - Created design documentation for the API(Alberto)
    - Created LeapMotion application for Windows(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for the removal of any Qt dependencies on the API
    - Added user stories for the mid-layer design of the API
    - Added user stories for the creation of the Kinect data application

Sprint 4 Report

Date: October 23, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Created GitHub documentation for future developers(Alberto)
    - Removed all Qt Dependencies from API GL Layer(Alberto)
    - Added Square shape drawing functionality to the API(Alberto)
    - Added Triangle shape drawing functionality to the API(Alberto)
    - Added Right Triangle shape drawing functionality to the API(Alberto)
    - Added Circle shape drawing functionality to the API(Alberto)
    - Created Kinect data application(David)
    - Created LeapMotion Documentation(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for shape compounding for the API
    - Added user stories for shape connection for the API
    - Added user stories for the development of a LeapMotion visualizer
    - Added user stories for the implementation of the Kinect visualizer

Sprint 5 Report

Date: November 6, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Introduced shape compounding for Right Triangles to the API(Alberto)
    - Introduced shape compounding for Lines to the API(Alberto)
    - Introduced Line connection for shapes to the API(Alberto)
    - Developed LeapMotion Visualizer and powered it with the API(Alberto)
    - Created Kinect Visualizer running with Kinect drawing API(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for unit testing of the API
    - Added user stories for 3D shapes introduction to the API
    - Added user stories to power the Kinect Visualizer using the API

Sprint 6 Report

Date: November 20, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
    - Developed unit tests for the API(Alberto)
    - Introduced Cube drawing functionality to the API (Alberto)
    - Introduced Cube compounding functionality to the API (Alberto)
    - Introduced Pyramid drawing functionality to the API (Alberto)
    - Introduced Pyramid compounding functionality to the API (Alberto)
    - Introduced shape transparency selection to the API(Alberto)
    - Introduced OpenGL Context color selection to the API(Alberto)
    - Updated Kinect Visualizer to run with TAMGeF API(David)
- Not Achieved:
    - N/A
- Product Backlog Changes:
    - Added user stories for the creation of UML documentation
    - Added user stories for 3D Shape color selection
    - Added user stories for the improvement of the LeapMotion visualizer

Sprint 7 Report

Date: November 20, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Achieved:
  - Improved API's performance(Alberto)
  - Fixed Circle aspect ratio bug in the API (Alberto)
  - Added color selection for 3D Shapes to the API (Alberto)
  - Improved Leapmotion's tracking capabilities by introducing finger tracking (Alberto/David)
  - Improved Leapmotion's tracking capabilities by introducing palm tracking (Alberto/David)
  - Added circles to the Kinect visualizer Skeleton joints(David)
- Not Achieved:
  - N/A
- Product Backlog Changes:
  - N/A

Appendix D - Sprint Retrospective Reports

Sprint 1 Retrospective

Date: September 11, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Both demonstrations went well, no issues displaying the work we have done.
- Reviewing the code base in order to find the bug that was preventing the visualizer from displaying any graphics was a difficult task(Alberto)
- Refreshing knowledge of C++ was not hard(David)

Sprint 2 Retrospective

Date: September 25, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Both demonstrations went well, no issues displaying the work we have done.
- Learning OpenGL programmable pipeline was a daunting task, and revising the codebase of the previous visualizer to implement it for all their drawing was complicated, but in the end it was achieved and did not produce any major difficulties (Alberto)
- Understanding how to design an API that is easy to use and is able to abstract the usage of many input devices took most of my time for this sprint. It was a very difficult task to achieve and required assistance of the product owner(Alberto)
- Implementing LeapMotion Console application in Mac OS X was not complicated(David)
- Migrating From OS X to Window and installing all the software required to accomplish the tasks was not that easy and straightforward(David)

Sprint 3 Retrospective

Date: October 9, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Both demonstrations went well, no issues displaying the work we have done.
- Devising an abstraction for the OpenGL Context creation currently running in the API was a difficult task as it required a lot of research and effort on my part(Alberto)
- Implementing the LeapMotion API in new system was easy(David)
- Creating console application for LeapMotion in Windows was simple(David)
- Implementing GUI LeapMotion application using Qt was curvesome, but was successfully achieved(David)

Sprint 4 Retrospective

Date: October 23, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Both demonstrations went well, no issues displaying the work we have done.
- Removing the Qt dependencies on the GL Layer of the API did not concur any major difficulties(Alberto)
- Devising an abstraction for the API to work with any kind of input device represented a major effort. In the end, the design decision was to implement an InputMapping class that would map any kind of coordinate system to GL coordinate system(Alberto)
- Introducing 2D Shape drawing functionality did not prove to be difficult(Alberto)
- Installing Kinect SDK was complicated because of the hardware requirements(David)
- Creating Kinect console application was simple(David)
- Creating Kinect GUI application was difficult and time consuming(David)

Sprint 5 Retrospective

Date: November 6, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Both demonstrations went well, no issues displaying the work we have done.
- Introducing Shape compounding to the API was a major task since I had to devise a way for the user to find it intuitive and easy to use. Developing the line connections, on the other hand, did not prove to be difficult(Alberto)
- Developing a LeapMotion Visualizer that would work with the API was not difficult but rather time consuming as I had to learn how to get the coordinates gathered by the device. The drawing however was very easy thanks to the API(Alberto)

- Drawing the Skeleton using Kinect was time consuming, but I gained experience(David)

Sprint 6 Retrospective

Date: November 20, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Demonstrations did not go that well. Diagrams were missing.
- Introducing 3D Shapes to the API was fairly simple now that I am more familiar with OpenGL. Implementing David's feedback for the API was also fairly simple(Alberto)
- Drawing Kinect skeleton using the API was very easy(David)
- Giving feedback for the API to Alberto was easy (David)

Sprint 7 Retrospective

Date: December 4, 2015
Attendees: Alberto Carrillo, David Escobar, Francisco Ortega
Discussed Topics:

- Alberto's demonstration went well.
- David's demonstration did not go as well since Dr. Sadjadi commented that David needed to explain his work with much more detail.
- Adding color selection to 3D shapes and fixing the aspect ratio bug of the circle did not prove to be difficult. It was a simple and rewarding task(Alberto)
- Working together for the first in the LeapMotion application went well. Introducing finger tracking and palm tracking to the LeapMotion was achieved through our combined effort(Alberto and David)
- Changing background color to Kinect visualizer using the API was easy(David)
- Giving different colors to Kinect skeleton using the API was easy(David)
- Drawing circle for the Joints with different colors using the API was easy(David)

REFERENCES

Some shaders for the 3D Shapes were taken from the following tutorial:
https://code.google.com/p/opengl-tutorial-org

Credit to John Ryding (strife25 in github) for offering a solution to drawing circles in OpenGL:
https://gist.github.com/strife25/803118