UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Neo Tandem Technologies

Eye Tracking Architectural Requirements

Author:	*Student number:*
Duran Cole	u13329414
Michael Nunes	u12104592
Molefe Molefe	u12260429
Tebogo Seshibe	u13181442
Timothy Snayers	u13397134

Centre for Geoinformation Science (CGIS)
University of Pretoria

October 28, 2015

Architecture requirements

Eye Tracking

Github Link: https://github.com/MichaelNunes/Neo-Tandem-Tech-Eye-Tracking

Version: 0.2 Basic requirements
Version: 0.5 Added content
Version: 0.9 Added in-depth content
Version: 1.0 Stable release (version 1) requirements
October 28, 2015

Contents

1 Architecture requirements

1.1 Quality Requirements

1.1.1 Scalability

The requirements for scalability in the program are that the storage of all the data will forever expand with the more uses of the program. This means that the longer you have the system the more data it will take up, this then states that we need to ensure that later on the space will not impact on the users' performance. The program itself can also grow in terms of features. This means that a developer can extend system by adding a new feature.

- Tactics and strategies
  - Compression: Compression of the files will allow more files to be stored on a hard drive. This can lead to greater scalability and improve space management.
- Integration: The integration of scalability in our system will allow the users to decide where the files that are written will be saved. This will enable them to save the files to hard-drives as time with the system goes on. This allows the user to expand their storage capacity as time goes on.
  The system is also created in such a way that everything is a module/component. The program allows for new modules to be created and then added. This encourages the growth of the system from the start as new and better components can be added very easily.

1.1.2 Performance

Performance of the system is key as the system is constantly reading in data as the eye is being tracked so the performance needs to be optimized so that the tracking of the eye does not stutter and then incorrectly track the eye. The system also needs to perform well and not cause issues when it is run.

- Tactics and strategies:
  - Thread pooling: this will allow us to spread the work load across threads so that they can run concurrently and reduce bottlenecks. This also allows us to optimize how we can use the hardware so that we use it to its maximum potential. The Eye Tribe SDK handles threading that is related to eye tracking by creating separate threads for gaze listeners.
- Integration: The system will be able to run the server that will get gaze data to run concurrently with the eye tracking function.Each of these is run on a thread and this allows them to run at the same time so all the data is collected in real time and the performance is optimized.

The system is spread into components which means that each component is responsible for a task and this increases performance as there are not waiting processes.

1.1.3 Maintainability

The system will need to be maintained over time as new technologies are introduced.The system is component based which makes maintainability easier as the components need to be maintained and not the entire system. Maintainability allows a program to be kept up-to-date. Changes in eye tracking is a common thing so we expect new changes to occur frequently.

- Tactics and strategies
  - Removal of Faulty Components: This allows us to remove components that are no longer working in the system and then replace them with a newer version of them all and if needs be a whole new type of component.
  - Deadlock Detection: The detection of deadlock is important and if a deadlock occurs could harm your system. This is important as detecting a deadlock can also then determine if the component is faulty.
- Integration: The system is component based so removing and maintaining the system is made easier.If we know that a certain part of the system is causing a problem such as deadlock or is faulty in any other way we can then decide to remove that component and fix it or add a new system. This way changes are made to a component and not the whole system.

1.1.4 Reliability and Availability

Reliability and Availability are important in this system. The server which is used for capturing the data needs to constantly be available to the program so that data can be read.The program also needs to be reliable and gather the information correctly. The collection of incorrect data can cause mishaps with the system. The system as a whole needs to be running and not crash and fail.

- Tactics and strategies
  - Load Balancing: By balancing the work load we can ensure that a system can continuously run and not have a crash. This call also cause services to constantly be available such as the server.
  - Removal of Faulty Components: This allows us to increase the reliability of the system by removing the cause of the unreliability.
- Integration: The load balancing can be achieved by separating the specific tasks into different components. This allows the system to share the load across different sub sections. The removal of the components can be done manually so that the system does not contain the faulty component.

1.1.5 Usability

When a user needs the system it will need to be usable, in other words the program should allow the user to explore all the features of the system and be able to use them all.

- Tactics and strategies
  - Contracts based decoupling: The system is decoupled by dividing the system into a set of contracts that each have its specification and have its pre- and post-conditions. This allows the system to be easily decoupled.
- Integration: The contracts created are used to make the system in to components which allows new contracts to be created as the system develops and there is no tight coupling amongst all the components hence there is no reliance on them.

1.1.6 Testability and Integrability

The system is quite a large system and the testing would need to be done on every aspect of the system this would mean that the system would need to have testing done all over. The system also needs to be integrated with all the systems components and this would mean that the system would need a high level of integration.

- Tactics and strategies
  - Layering: This will allow each component to be comprised of a layer and this layer can then be changed or removed. This allows good integration in the components.
- Integration: Each layer will have a set of components and these components can be changed and then integrated easily with the layers that are next to them. This allows the system to ave easy integration and allows changes to the system not effect the integration. The testing is made easy with each component allowed to be tested on individually so that the integration can be done seamlessly.

1.2 Integration and access channel requirements

1.2.1 Human Access Channel

The program will be accessible by human users via a desktop machine or a laptop that has and that can support the eye tracking equipment which requires a USB 3.0 (or higher) port for data transfer. The program is designed for and will only run any Windows operating system from Windows 7 onward.

1.2.2 System Access Channel

The System will be supported by any and all systems that support USB 3.0 (or higher) as well as run any Windows platform onward. The system also requires the machine to

contain Dot Net Framework 4.0 or higher to ensure program compatibility and efficiency. Another requirement will be that of internet access for the Eye Tribe recording server which requires it.

1.2.3 Integration Channel

The program will use a file system that stores all the eye tracking data in separate files. The files that are stored will then be used in the program to carry out functions such as making heat and gaze-point maps and giving result data to the user.

1.3 Architecture constraints

1.3.1 Hardware

The Eye Tribe eye tracking device will be used as the primary eye tracking recording device for this project. The Eye Tribe eye tracking device connects to the computer through a USB 3.0 port and as such it is required that the computer the application would be running on has USB 3.0 ports.

1.3.2 Operating System

The application only supports installation on computers running a Microsoft Windows operating system. The application may work on both 32-bit and 64-bit systems but 64-bit systems with large amounts of memory are recommended for memory processing.

1.3.3 Additional Software

You need to have the .Net 4.0 Framework installed or better to run the program. This ensures that all the current technologies are used properly and to the best of their ability. The Eye tribe SDK will also need to be installed so that the camera can be calibrated and data can be transferred to the program.

1.3.4 Version Control Management

GitHub will be used to as the version control management system for the duration of this project. This is a distributed version control meaning that every node has a copy of the repository. With this tool we can easily track issues and bug fixes throughout the program as well as manage new features and testing with the use of branches.

1.3.5 Development Environment

The application will be programmed in the C sharp(C#) programming language through Microsoft Visual Studio 2013 Community Edition. This is a free version of Microsoft's Visual studio and is built for small development teams such as up to five members. The

additional software (namely The Eye Tribe SDK) being used also provides C# libraries that can be used to access additional features and capabilities of their software and hardware.

2 Architectural patterns or styles

2.1 Singleton Recorder

The program uses a server to gather and record the data from the eye tracking camera. This server is constantly running and there should only be one instance of this class running so that we do not have multiple servers running and all the data is pulled consistently from one location. The server will thus need to follow the singleton design pattern. The singleton design pattern is a creational design pattern that is used when only one instance of the object should be created for the application. This will provide the solution of only one server object being created at time. It will check whether an instance of the requested object is created and if it is the object is referenced and not recreated and therefore there can only be one instance of the class created.

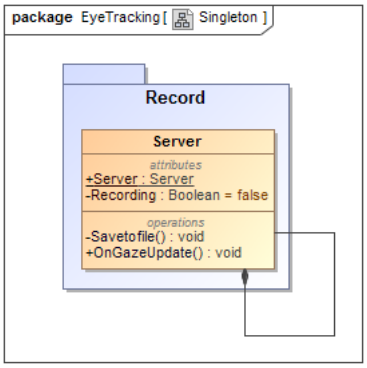

Figure 1: Singleton Recorder

2.2 Flyweight Settings

The program has data that needs to be shared amongst modules and that also needs to be used through-out the system. This is achieved by use of the flyweight pattern. The application requires all system and model settings to be created, saved and eventually used by the rest of the modules within the system. These variables are static and are defined as internal so that only the program can access the stored setting values. This ensures that all the modules are able to read and change the shared memory resource.

2.3 Composite

With in our program we have built a graphics module with the help of OpenTK that allows us to render three-dimensional (3D) models, preform actions on them and return

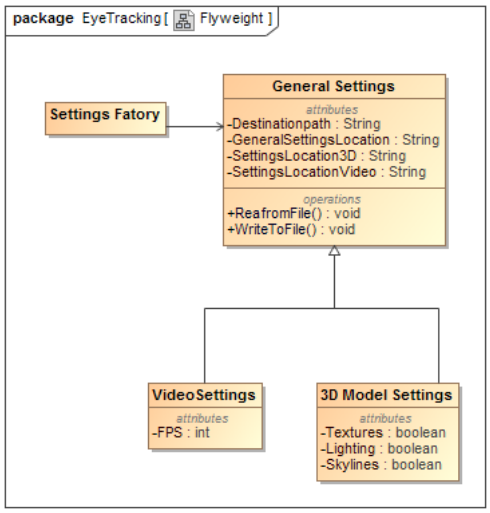

Figure 2: Flyweight Settings

desired results. When it comes to building the model we first have to import it and start the process of parsing the model into data we can use to create a visual representation. Each model is saved in an object file (".obj") this file contains a parent game-object and its multiple children game-objects. This pattern creates a tree of each game-objects and its sub-game-objects.

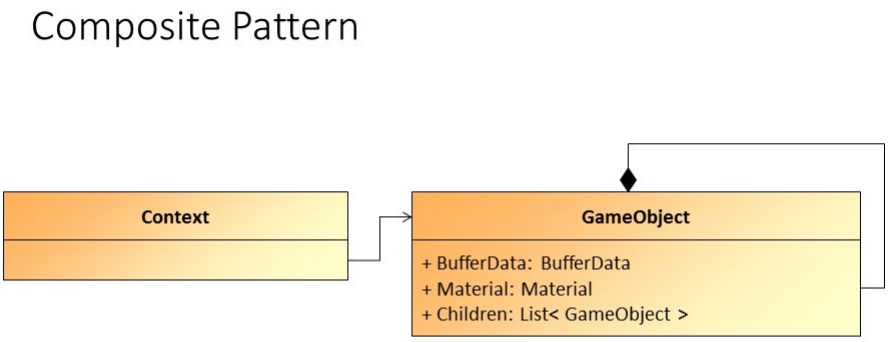

Figure 3: Composite Game-Objects

2.4 Private Class

This pattern gives the system a means of separating all game-objects and their attributes into different containers and thus aids in simplifying the class definition.

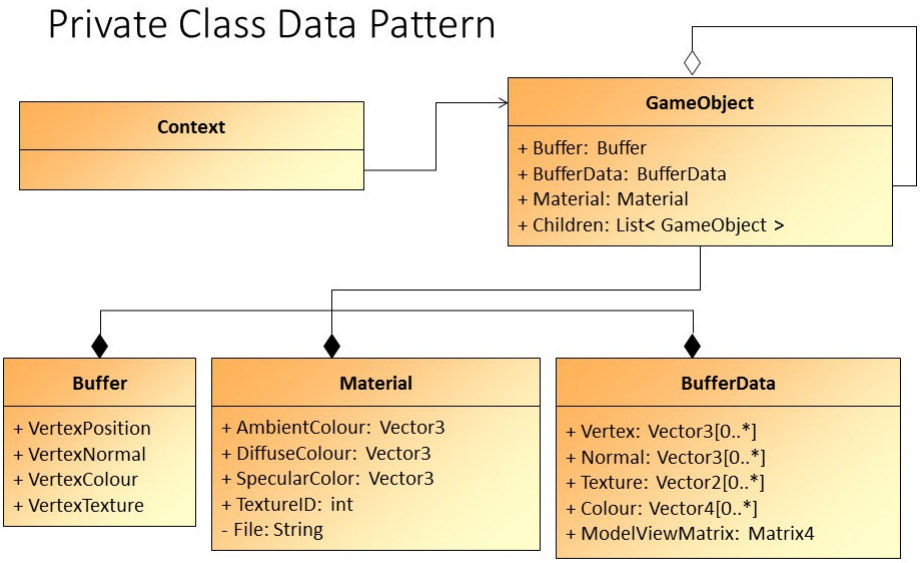

Figure 4: Private Class Game-Objects

2.5 Facade

When you wish to import and view a model the first action that will occur is a call to the display model class. The facade will call a converter to parse the object file and call the game window to render the object. This pattern hides the complexity of both generation and rendering the model by just providing a single, simple entry point.

2.6 Template Method

With in the space of three-dimensional models multiple features can be applied, such as different lighting techniques. The pattern consists of the base light class which is an abstraction of its three sub-classes. In the system we make use of this pattern to decide what type of lighting will be applied to a model. There are three types of lighting, namely: ambient, point and directional lighting. A combination of any three can be used but currently all three are being utilised.

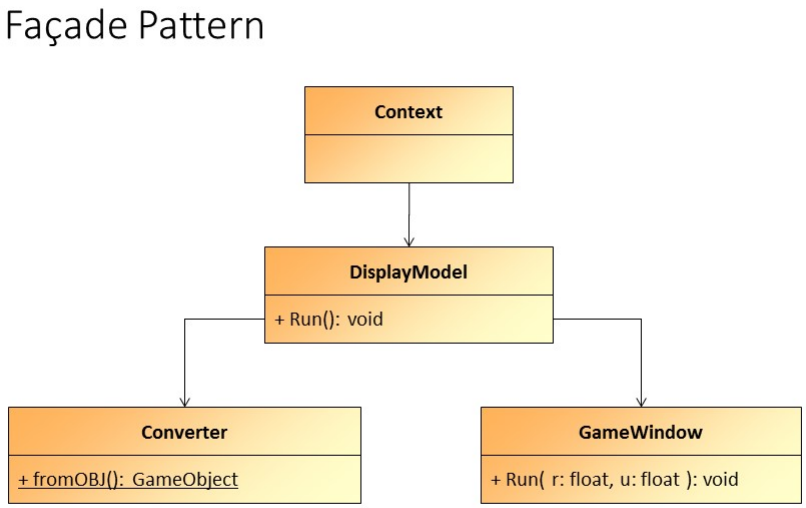

Figure 5: Facade Game-Objects

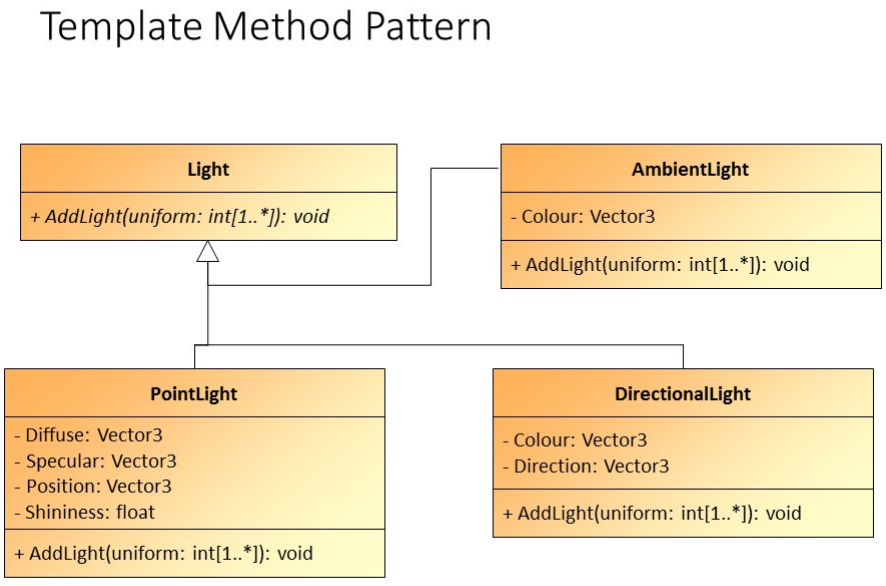

Figure 6: Template Method Game-Objects

2.7 Builder

As multiple different three dimensional objects models can be imported all with different structures and objects, we make use of the builder design pattern to separate the

construction of each game-objects so that each object can be built dynamically based on its requirements. A three dimensional object consists of a buffer, vertex buffer objects, materials and a list of children objects. When parsing the parsing the ".obj" file all objects are created, followed by the parsing of the ".mtl" file where all materials are created and attached to the objects. The root object and its tree structure is then generated and returned.

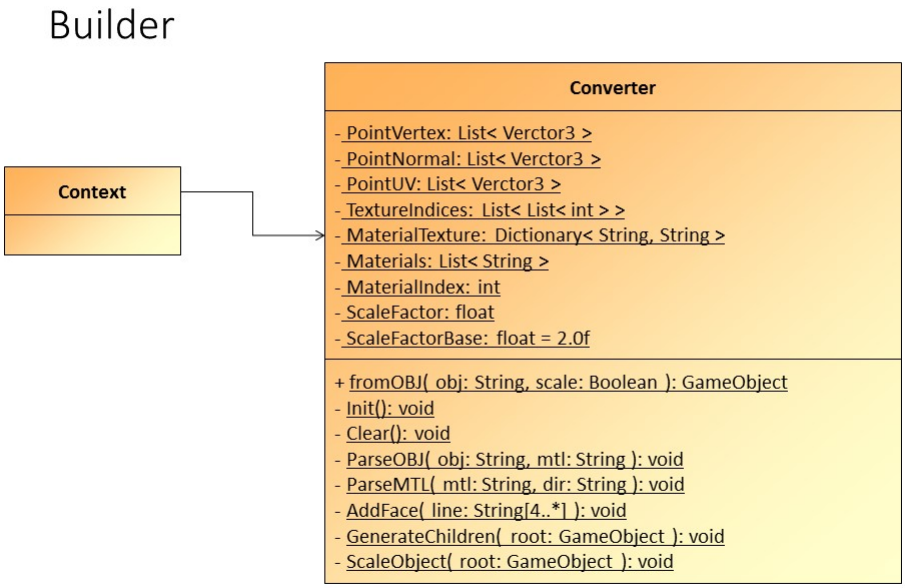

Figure 7: Builder Game-Objects

3 Architectural tactics or strategies

3.1 Availability

This refers to whether the user will have constant access to the program and methods to avoid interruptions.

Because of the way in which the system will work, all aspects of the systems will be working as long as it is running. Therefore, an exception based fault detection system is required to ensure nothing fails. Unless it is the user interface that fails, the exception will be handle the error in the background, and a connection between the sections of the systems will be re-attempted.

Checkpoints will be saved in order to roll back in the event of errors.

3.2 Scalability

This refers to time and cost relating to development, changes, and testing of the system.

The MVC pattern allows us to separate concerns of the system allowing for finer grain implementation and changes to its inner working. This will also allow for potential changes to be "plugged in" if required.

This is also means that testing will be easy as each section simply needs to test whether they are able to do their own required task. All related task will simply be using the abstraction of the subsystem.

3.3 Performance

This refers to the speed of execution, the time between a request and a response between modules within the system.

In order to simplify the subsystems, any optimized, pre-existing algorithms that can help with their executions will be used. The least possible intermediaries will be used. Concurrency will also be used in sections to ensure speed of generation when it comes to creating heat maps and gaze plot.

3.4 Testability

This refers to the ease of testing of the system, during successive build releases.

During each build release, all test cases used in the subsystems unit tests, will be combined into a series of big test cases in order to make sure that the expected input provides the same expected output.

3.5 Usability

This refers to the ease of use the user will have with regards to the system.

Since the system will be doing most of the work, the user will only be provided with a very simple interface to interact with the basic functions i.e. record screen, print to texture.

4 Use of reference architectures and frameworks

The application will be coded in C# and thus we will be using the relevant architectures and frameworks to create this application.

4.0.1 .Net Framework version 4

This framework was developed by Microsoft and is an integral part in C# development, as it forms the basis of the language and its capabilities. All our core functionality is written around functionality designed and provided by this framework.

This framework was developed by Microsoft and is an integral part in C# development, as it forms the basis of the language and its capabilities.

4.0.2 NUnit

This is a unit testing framework developed for all .Net languages, and in turn takes advantage of a lot of the languages capabilities. This will allow us to create tests and run them on specific functions without having to run the full application to test functionality. This is beneficial in that we can test functions before finally integrating them in to the final system.

4.0.3 Eye Tribe SDK

This forms the basis of our application as this provides the ability to record and interpret where the user is looking and feed real time information that can then be used to build heat-maps and statistics.

4.0.4 HeatMap.NET

This provides the functionality to build heat-maps based on information that is fed to the module. Information provided to this heat-map module will be that of the recorded eye tracking data which will create the physical heat-maps for the users.

4.0.5 aForge.NET

This library provides us with in depth media options especially with regards to the video media type, allowing us to be able to get almost all meta-data from an imported video as well as take the imported video and split it into a frame per second image library to which our heat-maps can be applied. It also provides the functionality to create videos out of these newly created images.

5 Technologies

We have chosen the following technologies carefully as they incorporate into one another to form a sound basis for our project to operate on.

- **C#**: C# is a programming language that was designed and developed by Microsoft.The language is a object orientated language and is highly focused on component-oriented. C# is used with the .NET frame work which was discussed in a earlier section. The language is a very versatile language that is used when developing most Microsoft projects. The language is able to create classes and a GUI which allows us to map certain functions of the GUI to the class.

- **The Eye Tribe eye-tracker**: This is one of the cheapest if not the cheapest eye tracking cameras available on the market.The tracker is infra-red eye tracker which can track gaze data and then this data can be used to carry out numerous functions.The Eye tribe eye tracker can be used on a computer or a mobile device such as a smart phone.The eye tracker is $99 eye tracker that makes it cheap and competitive with the more expensive ones on the market.

- **Eye tribe SDK**: This is a full SDK that is used to develop programs that will use the Eye-tribe tracking camera. The SDK comes with the user interface and server components to help design new programs that are compatible with the eye tribe software.

- **GitHub**: This is a version control system that is used to manage the versions of software. Github uses a copy-modify-merge system where the users will clone a repository and then modify it and then merge it. It is a distributed version control so every person has a copy of the repository.

- **NuGet**: This is a packet manager which helps to manage all the packages in a project. This allows users to add packages from all sources to their projects to use certain parts of it. This also manages the packages so that you don't have to worry about locating them and referencing them.

- **Nunit**: This is a testing framework built for the .Net framework.This is very similar to Java's JUnit. The same premise and method is used to create and execute tests. The tests will show a positive result if they have passed. If a failure occurs on a test it will tell you where and what cause the error. NUnit is used to write unit tests so that each part of a program is tested.

- **OpenTK**: OpenTK or Open Toolkit Library is a OpenGL library that allows the users to manipulate 3D models inside a .Net projects such as a C# project. This means that a 3D model can then be manipulated and used in a program.

- **Visual Studio Community 2013**: Visual Studio Community 2013 is a robust IDE used to write .NET projects such as a C# project. The IDE has many features that help improve development and increase productivity. The Visual Studio community edition is free to developers if you sign up with a Microsoft account.

Bibliography

[1] Microsoft (2015), *.Net Framework*. Available: www.microsoft.com/net

[2] NUnit (2015), *NUnit*. Available: http://nunit.org/

[3] The Eye Tribe (2015), *Eye Tribe SDK*. Available: https://theeyetribe.com/

[4] NuGet (2015), *NuGet*. Available: https://www.nuget.org/

[5] OpenTK (2015), *OpenTK*. Available: http://www.opentk.com/

[6] aForge.Net (2015), *aForge.Net*. Available: http:http://www.aforgenet.com/