**Vi-Char**

**Motion Capture Division**

**Intermediate Report**

Daniel Porter

Ian Earle

Travis Dos Santos-Tam

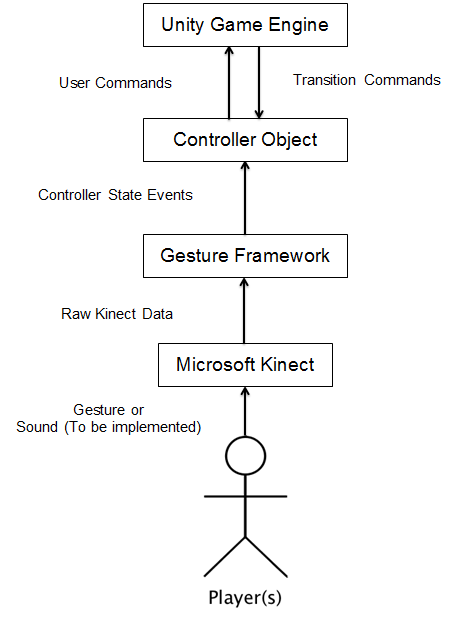
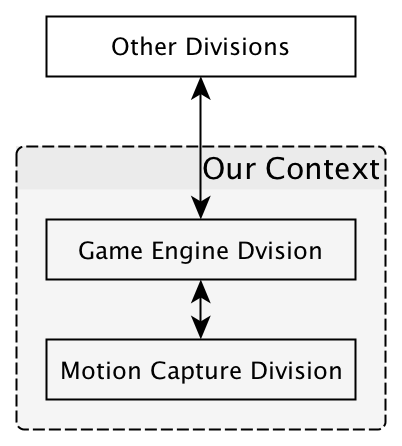
Max Gobel

**Abstract:**

The Vi-Char system is incorporating a number of different technologies for a wide variety of purposes. Within this system, motion capture is being used as a method of input, and so the system is to be designed in a way as to best facilitate a smooth and easy to understand user experience. The motion capture tools act as one of the primary ways which the users interact with the system at large, and it is our goal as a group to create a product which serves as a seamless interface to the user while efficiently providing all of the required information to the Game Engine.

This report will elaborate on the present implementation details of the Vi-Char Kinect Controller system, and explain some of the design choices that went into them. Emphasis will be made on the system’s ease of inclusion into an existing code base, adaptability, and maintainability.

**Design Summary**



As illustrated above, the internal workings of our system will be very isolated from the rest of the group. The idea behind our design was to allow the Game Engine group to transition their control scheme with as little refactoring as possible. This loose coupling is allowed by the creation of the “Controller Object”, which contains several “Controller States.” Each controller state has an action associated with it from the perspective of the Game Engine when active, and each state has a Gesture by which the user can activate it.

**Install Documentation**

For the Intermediate Demo, installation is as simple as opening the solution in Visual Studio (tested in 2010 - likely usable in 2012) and running the “Console Application 1” project.

-To create your own gestures, open up the GestureMap.cs file in the Gesture Framework file. See the commented out “template” example. The gesture\_name field is not in displaying the gesture, so if you want to see a new kind of Gesture “reported”, add the gesture type to the GestureType enum, also in the GestureMap.cs file.

-The Gesture vocabulary is incredibly flexible, and allows for a wide variety of gestures to be expressed. It should be noted that the accuracy of Front/Behind is not perfect, due to the nature of the Kinect’s mapping, but still displays a relatively high level of accuracy.

For inclusion of the Controller into an existing project, simply include either the source into a C# project, or acquire the corresponding DLLs and include them into an existing project. In either case, once the appropriate files have been included into your project, using the Controller\_Core namespace, create a Controller object. Once the object has been created, you can access it’s properties (Moving, Jumping, Turning - or Define your own!) to determine the present state of the controller.

Source Code is available at UPS-CS240-F12 - Motion Capture - Dev repository presently. Pre-compiled DLLs are also available by request.

**Implementation Details**

**Event Driven Control Flow**

When the Controller object is initialized, it starts up the Kinect and registers several event handlers. Once that has occurred, the Controller object becomes a passive receiver and presenter of information.

Whenever the Kinect has prepared a skeleton frame, it fires off an event alerting the Gesture Framework within the Controller to check the frame for a Gesture Component. If such a component is discovered, it checks to see if it completes the gesture. If so, it fires off another event to update the appropriate Controller State. Otherwise, it waits for the next event.

The advantage an Event Driven flow model is that no thread management has to be done and it integrates with the Kinect’s behavior patterns most smoothly.

**Controller-Based Command Passing**

Instead of continuing to pass the Gesture Events all the way up to the Game Engine, we opted to have all Gesture Events update their corresponding Controller State. While this will require some tuning of the duration of each activation for any project the Kinect Controller is included in, the system is flexible enough to be easily tuned. This also prevents the user from having to adopt an Event Driven Control Flow. This will be particularly when integrating with the Game Engine group, which has structured their Game to expect Controller updates every frame. Since it is unlikely that the Kinect will be receiving 30+ Gestures a second, giving the Game Engine group a stable interface to regularly check for updates is much simpler than wiring them into the Event Chain.

**Flexible Gesture Recognition**

Given the organic nature of Gesture recognition, we opted for a Gesture Framework that was easily adaptable, as well as one that was able to account for a wide variation of appropriate motions. The Coding4Fun Framework (with some adaptation) fit the bill.

Firstly, the Framework allows for easy Gesture definition with a simple “vocabulary” of Gesture Components. Each Gesture consists of one or more of these Components, and each Component has two joints associated with it, as well as at one or two Joint Relationships. All of the Components of a Gesture must occur within the time limit after the first component is recognized for the Gesture to be accepted.

A Joint Relationship describes the relative positioning the two joints must have with respect to each other for a particular Component to be accepted. Example Relationships include: Above, Behind, LeftOf, BehindLeftOf, BehindAboveAndLeftOf, etc. Joint Relationships can evaluate all three axes, however, if an axis is not included in the Relationship, it will be ignored when evaluating the Component. For example, LeftOf ignores the Y and Z planes, and only evaluates if Joint 1’s X value is lower than Joint 2’s.

Secondly, the Framework needs to allow for a wide variety of Player motions to be accepted as the same Gesture. The Joint Relationship evaluation described above assists with this as well, since any motion which satisfies the relative positioning requirements of the Component’s Joint Relationships will be accepted.

**Procedures Followed**

**Source Code Management:**

GitHub/Visual Studio Built-ins

Google Docs for document management

**Testing and Verification:**

Gestures were described verbally in a sentence or less (without gesticulation) to housemates/roommates unassociated with the project. If the test subject was able to easily get the Gesture to register with a minimal amount of flailing, we considered that the corresponding Joint Relationships in the Gesture Framework were acceptable.

**Coding Style:**

Public Methods and Properties: Uppercase first letter of name

Private Methods and Fields: Lowercase first letter of name

All Kinect related variables should include “sensor” in the name.

-Exception: Any SDK code will be left as is when possible.

Comment Conventions: Single line comment unless otherwise necessary. Naming of parameters should be clear enough that it is unnecessary to describe them. If not possible, do so in a C# ///-summary block.

**Reflection**

Thus far, our largest challenge has been overcoming the learning curve of developing with the Kinect. Given the complexity of the peripheral, there is an immense body of literature and software regarding developing for it. Unfortunately, given the immensity of that body, finding the “diamonds in the rough” is not always the most straightforward. Fortunately, Visual Studio’s auto-complete and debugging features were robust enough to eventually give some insight into the process of handling the Kinect’s raw output.

**Timeline**

1. Establish version control system (Sept 17th-19th) - Completed October 13th

2. Setup Kinect development environment (Sept 19th-24th) - Completed September 19th

3. Begin work on Kinect Motion-To-Event Translator (Sept 24th-Oct 22nd) Completed October 21st

4. Discuss Events to be contained within the shared Control Dictionary with Game Engine Group (Oct 22nd-Oct 29th) - Pending

5. Tuning of Gestures based on input from the Game Engine Group (Oct 29th-Nov 12th) - Pending

5. Integrate Game State Management with the Game Engine Group. (Nov 12th-Nov 19th) - Pending

6. Finalize Gesture Vocabulary and run through User Acceptance Testing. (Nov 19th-Nov 24th) - Pending

**Glossary and References**

**Glossary:**

-**Control Dictionary:** A set of events shared with the game engine, which are mapped to a set of player actions. Upon perceiving the appropriate player action, the Kinect will send the corresponding event to the game engine for processing.

-**Controller State:** An attribute of a controller object, each controller state presents information regarding whether or not it is active, and what type of action it should be associated with.

[**References:**](http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series)

**Included in the Project:**

-Base for the Gesture Framework

(<http://channel9.msdn.com/coding4fun/kinect/A-Simple-Gesture-Processing-Framework-for-the-Kinect-For-Windows)>

**Other References:**

-Microsoft [Kinect for Windows SDK (](http://developkinect.com/)[http://www.microsoft.com/en-us/kinectforwind](http://www.microsoft.com/en-us/kinectforwindows/%29)[ows/)](http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series)

[-Kinect Dev Tutorials (http://www.kinecthacks.com/kinect-sdk/)](http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series)

[-Zigfu Development Kit (http://zigfu.com/)](http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series)

[-Possibly useful for in](http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series)tegrating with the game engine

-Develop Kinect Community (<http://developkinect.com/>)

-Getting Started with Kinect Development (<http://channel9.msdn.com/coding4fun/kinect/Getting-started-with-Kinect-development-quickly-with-the-Kinect-for-Windows-Quickstart-Series>)