Elemental Kinection

Developer Guide

Version 2.0

2 May 2016

Revision History

All revision history listed below.

Version	Change Summary	Date
1.0	• Initial Draft	23 April 2016
1.1	• Minor formatting	2 May 2016

Revision Sign Off

The following asserts that all team members have read the document and assert that the information contained within this document is complete and correct.

Name	Signature	Date
Samuel Kent		
Jack Kempner		
Nathan Johnson		
Aakash Tyagi		

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to provide future developers of the Elemental Kinection project a reference for the tools used, structure of the project components, and the environment for development. This information is for use in the event of Elemental Kinection's continued development.

1.2 Overview

Section 2 – System Overview
Section 3 – Desktop Application Development
Section 4 – Web Application Development
Section 5 – Database Development
Section 6 – Glossary of Terms

2 System Overview

2.1 System Components

The Elemental Kinection project is divided into two main parts, the **Desktop Application**, and the **Web Application**. These two applications are in turn supported by a **MySQL Database** and **Microsoft SDK Tools**

2.2 Desktop Application

The Desktop Application is built within the Unity game engine with a combination of graphical elements and C# scripts. It connects with the Microsoft Kinect v2 to track patients' exercises.

2.3 Web Application

The Web Application utilizes a series of tools and frameworks. The web server is hosted on an Amazon Web Services and designed using the Python framework Django. Nginx (pronounced EN-JIN-EX) handles load balancing and other monitoring tools.

2.4 MySQL Database

MySQL is an open source database management system. It provides a way for interacting with relational databases. In our project, the MySQL database holds all of the therapist, patient, session and result data.

2.5 Microsoft SDK Tools

Two tools in particular are used in this project. The first is Microsoft Visual Gesture Builder, a tool for defining gesture recognition files. The second is Kinect Studio, a tool for recording different Kinect data streams.

3 Desktop Application Development

3.1 Environment Setup

This project was developed in Unity 3D.

There are two tiers of Unity: Personal and Professional. For this project Personal Edition has all the features necessary. The download is available for Windows and Mac OSX. This particular project was originally developed on the Windows version.

The version of Unity used to develop the project was 5.2.1; however, the most current version of Unity is 5.3.4 with 5.4 in Beta. You can download archival versions of Unity at https://unity3d.com/get-unity/download/archive. There is always the possibility of downloading a newer version of Unity will not be backward compatible with the features used in this project.

Unity uses C# and/or JavaScript scripting to add functionality to the scenes in the project. You can choose to use either Visual Studio or MonoDevelop. The original development of the project was done in Visual Studio and all the scripts were written in C#. Information on how to integrate Visual Studio and Unity can be found at https://docs.unity3d.com/Manual/VisualStudioIntegration.html.

Finally there is plugin that must be added to Unity in order to use the Kinect and Visual Gesture Builder Frames. At the time of writing, the file with the necessary plugin can be found at http://go.microsoft.com/fwlink/?LinkId=513177. There are also assets in the Unity store that can potentially help in setting up the environment although not used in our project.

3.2 UI

The UI is very similar to Java where UI elements, such as buttons, can be placed on to panels which can in turn be placed on a canvas. It is important to note that depending on what level of the hierarchy an element is, a different relative coordinate system will be used. This will affect how your scenes appear visually in the final build. Pay attention to the hierarchy as some UI elements, such as buttons, will come with children components that will need to be accessed to change important values.

3.3 Scripting

Unity compiles the C# scripts created by the programmer using the C# compiler included with Mono 2.0. Mono is an open source implementation of the C# compiler and common language runtime. Mono is behind the most current version of .NET and the version of Mono used by Unity is behind the most current version of Mono. This chain of lag results in a development environment

similar to that of .NET 2.0 with some 3.5. This will limit some aspects of programming with the Kinect especially when dealing with newer data structures that the gesture builder class uses.

3.4 Terrain

Unity has many tools both built-in and available on the Asset Store for creating engaging background terrain. One main point to be made here is that the more materials and prefabs in a project, the bigger the project file and more demanding the application becomes. Grass in particular can be a big performance hindrance.

3.5 Important Scenes

The following are essential screens of the Desktop Application.

3.5.1 Login Scene

This screen is labeled **UI** in the project. It contains basic scenery and the login UI. Most of the important scripts are attached to the Canvas Object.

3.5.2 Session Scene

This scene is labeled **MainScreen** and contains some scenery. The scripts are also kept in the Canvas object with special attention paid to Generate Exercises, which will be covered in more depth later.

3.5.3 Introduction Scene

This scene is called **info scene** and has a pleasant forest scene. The important script, called Faded, is kept in the ‘control text’ object.

3.5.4 Exercise Scene

This scene is called **repcountscene**. It contains many important objects. ColorView handles the color video display that appears on the white screen along with ColorSrcManager. The BodySourceManager is responsible for tracking the patient’s body and Body view tracks the body and displays the skeleton.

3.5.5 Statistics Scene

This scene is called **statistics**. The big script in this scene is Grapher which is located on the Canvas object. You may notice some UI elements off the screen. They are either positioned there for the sake of final build appearance or to act as a template that other objects will be generated from.

3.6 Important Classes

There are many classes in the Desktop Application, but the following are especially crucial.

3.6.1 Gesture_Recognition

This class is responsible for loading the gbd file, tracking the patient, and recognizing when a gesture has been completed. The start() method contains all the actions that occur when the scene first starts. This includes setting up the UI elements and populating them with the correct reps and time information. It then proceeds to start up the Kinect, look for the appropriate gbd file, and either open the file or download() the required file from the website. The gesture recognition occurs from roughly line 265 on. There are three ways to exit the scene: completing the reps, running out of time or hitting the exit button. These three action listener stubs behave very similarly in accessing the database and uploading total number of repetitions and time elapsed for an exercise to the results table.

3.6.2 generateExercises

In the start() method, the data from the sessions table is downloaded. The exercise buttons and accompanying labels are created from templates and populated with data from the database. Write_sess() and write_exers() are currently unused, but can be used to save local results data to the PC. There are 4 different exercise statuses that are commented on in further detail in the code.

3.6.3 Grapher

Grapher is best explained in the logical progression of its use.
1. It downloads all the unique exercises and unique session due dates for that patient.
2. It uses that information to populate a list of exercises along the left and the date selection drop down lists.
3. Action listeners are attached to the exercise buttons then the buttons are disabled
4. Action listeners are attached to the mode buttons then they are disabled.
5. The update() waits for a valid range of dates to be selected from the dropdown lists. Then enables the mode buttons.
6. When one of the mode buttons is selected the toggle variable is switched to represent the mode and any current data being displayed in the graph is destroyed. The exercise buttons are also enabled.
7. When one of the exercise buttons is pressed all the results for that range of dates is retrieved from the database.
8. Depending on the value of toggle reps or time data is displayed.

3.6.4 PersisInfo

This class persists through every scene and holds information retrieved from the database in the generateExercises class for later use. It has getter and setter methods and could be conceptually thought as a global variable class for the entire application. This is required due to Unity not using traditional global variables.

3.7 Kinect Development

As mentioned above, the Kinect plug-in for Unity from Microsoft provides the libraries and classes for development of the Kinect in Unity. There are some methods that are different from the standard Kinect development for Windows. An example is the VisualGestureBuilderDatabase constructor. In Windows to open a VGB database the format VisualGestureBuilderDatabase(path) is used, but in Unity it is VisualGestureBuilderDatabase.Create(path). This alteration is reflected in the source code.

4 Web Application Development

This project's source code is hosted on a Bitbucket repository which can be accessed at:
https://bitbucket.org/aakashtyagi/senior-design-kinect
The project directory is divided into 2 main parts:

1. **elemental_kinection**
   This is the folder consists of project's urls and settings in urls.py and settings.py files respectively. To make any changes in the settings related to database connections, media servers, or pre-defined paths for static files and folders, changes should be made to the settings.py file under elemental_kinection.
2. **kinect_app**
   This is the main project folder which consists of all the database models, views, forms, urls, and HTML templates.
   a. models.py – This file consists of all the database models. Database tables are defined as individual classes.
   b. forms.py – This file consists of the forms used in HTML templates to collect the information from the user.
   c. views.py - This file consists of behind-the-scene functions or the backend of the web application. To add a functionality, define a new function in this file based on the requirements.
   d. urls.py – This file consists of the urls used to navigate the web application.
   e. templates – This folder consists of the HTML templates used on the web application.

Using Git, you can fork this project on your local computer by using this command:
https://aakashtyagi@bitbucket.org/aakashtyagi/senior-design-kinect.git
If you do not have Git, it can be downloaded from: https://git-scm.com/

4.1 Environment Setup

This is a Django-based web application. To run this project either on a local server or on a production server, there are some pre-requisites you need to have setup in the desired environment. These will get you up and running in Django:

1. Install Python
   As it is a Python Web framework, Django requires Python. This project uses Python 2.7.x which can be found at https://www.python.org/downloads/
2. Get your database running
   If you plan to use Django's database API functionality, you will need to make sure a database server is running. Django supports many different database servers and is officially supported with PostgreSQL, MySQL, Oracle and SQLite.
   This project uses MySQL as its database, which can be found at
   https://pypi.python.org/pypi/MySQL-python

3. Install Django
   This project uses Django 1.7.9. To install Django, using 'pip' is recommended. Pip is a package management system used to install and manage software packages written in Python. You can get pip at https://pip.pypa.io/en/stable/
   After pip is installed, open your command line and execute:
   "pip install Django==1.7.9"
4. Django Bootstrap Form
   Django Bootstrap Form is a python library that implements bootstrap structure to Django forms. This can be downloaded from https://github.com/tzangms/django-bootstrap-form
5. Django chartit
   Django chartit is the graphing library used to generate the results graphs on the web application. This can be downloaded from http://chartit.shutupandship.com/

Once you have everything installed and working on your computer, open your command line and execute these commands:

1. **python manage.py makemigrations** (this command is accountable for any new changes made to the database)
2. **python manage.py migrate** (this command makes the changes to the database models and syncs them with the web app)
3. **python manage.py runserver** (this commands starts your localhost and shows up your web application at 127.0.0.1:8000 in your web browser)

Note: These commands must be executed whenever any change is made to the database models.

4.2 Production Server

This app is hosted on Amazon Web Services (AWS). To ssh into the instance, download puTTy (http://www.putty.org/), a handy tool that allows you to ssh into remote servers (Windows users).
SSH details are as follows:
Host name: ec2-52-36-159-123.us-west-2.compute.amazonaws.com
Port: 22
Connection type: SSH
Website URL: ec2-52-36-159-123.us-west-2.compute.amazonaws.com:8000
Expand SSH under Connection in the left hand bar, and click Auth. Click browse and select the file "new-key.ppk" and click open. You will find this file in your project folder that you forked earlier from Bitbucket repository. This is the authentication key.
When prompted for username, enter "ec2-user". You can now navigate to project directory by executing this command:
cd proj1/senior-design-project/
Now setup the environment the same way as described earlier.

If you are a Linux user, follow this official AWS guide:
http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html

Note: Whenever you make any changes to the files, to reflect the changes on the website, execute the following commands in sequence:

1. **sudo service nginx restart** (to restart the nginx services)
2. **uwsgi --ini uwsgi.ini > log/elemental_kinection.uwsgi.o 2> log/elemental_kinection.uwsgi.e &** (restarting the Django wsgi script using uwsgi)

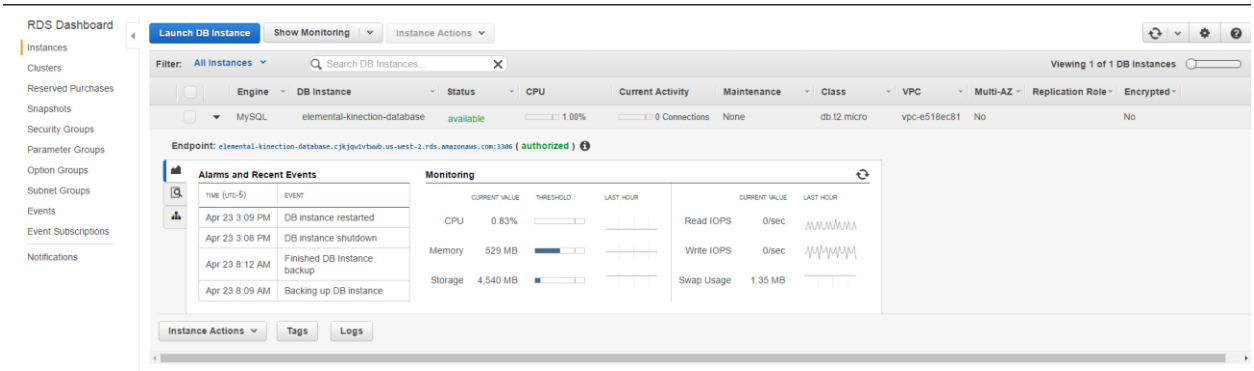
5 Database Development

5.1 Overview

The database of Elemental Kinection is a MySQL database hosted on Amazon Web Services using their RDS service. It stores all of the therapist information along with patient results and session data. A combination of the AWS RDS Dashboard and Django administration page are used to manage the database.

5.2 Setup and Administration

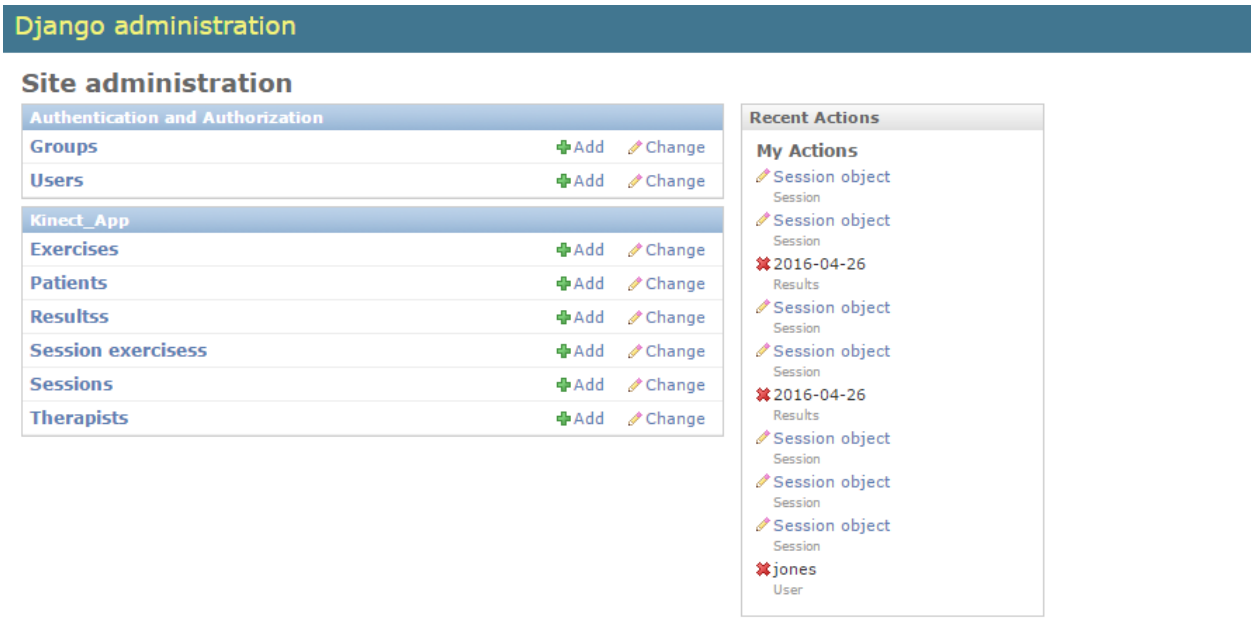
To access the AWS dashboard input your Amazon credentials on the AWS site. Once authenticated, you can navigate to the RDB service where you will find the “elemental-kinection-database” instance under the “Instances” tab. You should be able to monitor the status, number of connections, load, and storage of the database.



Home Page

You can reboot the database from here if it needs to be refreshed. Rebooting the database does not delete any of the data, but will end any current connections. The “Launch DB Instance” allows for the quick creation of another database if necessary.

Navigating to http://ec2-52-36-159-123.us-west-2.compute.amazonaws.com:8000/admin/ will take you to the admin page for Django. From here you can see the tables in the application and modify data with the graphical interface provided.



Django Admin Page

5.3 Accessing the Database

You will first have to download the MySQL connector from
https://dev.mysql.com/downloads/connector/net/
This should give you the MySql.Data.dll, System.Data.dll and System.Drawing.dll that you need to access the database. The connections string will look similar to the following:

```
public string conn_string = "Server=elemental-kinection-database.cjkjqw1vtwwb.us-west-2.rds.amazonaws.com;Port=3306;convert zero datetime=True;Allow Zero Datetime=true;Database=elemental_kinection_database;Uid=user_name;Pwd=password;";
```

From there you should be able to follow the standard C# implementation of opening a MySQL connection and executing statements.

5.4 Database Schema

Patients

This table stores the information for patient accounts. No personal information about the patient is stored in the database.

Attribute Name	Attribute Description
id	Primary key, used to uniquely identify a given patient in the database.
therapistId_id	Foreign key, referencing the therapist that is currently presiding over therapy sessions for the specific patient.
username	Used as part of the patient validation process, when the patient logs into the website or the Kinect application.
password	Used as part of the patient validation process, when the patient logs into the website or the Kinect application.
start_date	The starting date of the patient's therapy.
end_date	The ending date of the patient's therapy

Therapist

This table stores the information for therapist accounts.

Attribute Name	Attribute Description
id	Primary key, used to uniquely identify a given therapist in the database.
username	A string that is used as part of the validation process, when the therapist logs into the website.
password	A hashed string that is part of the validation process, when the therapist logs into the website.
firstname	First name of the therapist.
lastname	Last name of the therapist.
workplace	The therapist's place of work.

Admin

This table stores the information for admin accounts.

Attribute Name	Attribute Description
id	Primary key, used to uniquely identify a given patient in the database.
username	A string that is used as part of the validation process, when the admin logs into the website.
password	A hashed string that is part of the validation process, when the admin logs into the website.
lastname	Last name of the Administrator.
firstname	First name of the Administrator.

Session

This table stores the information for the patient therapy sessions.

Attribute Name	Attribute Description
Id	Primary key, used to uniquely identify a given session in the database.
ordering	Auto-incrementing field used to denote sequence of exercises in a
sessdate	Date the session this exercise is assigned to is due.
exercise	Foreign key, references the exercise table
reps	Total reps needed for this exercise
time	Time allotted to complete the exercise in seconds.
completed_time	Time elapsed for completion of this exercise.
finished	A status field for state of exercise. 0 = not done, not sequenced 1 = done 2 = not done, sequenced, available 3 = not done, sequenced, not available
user	Foreign key, referencing the patient to whom this therapy session is

Exercises

This table stores the exercises that the program supports.

Attribute Name	Attribute Description
id	Primary key, used to uniquely identify a given exercise in the
exerciseName	A string that contains the exercise name.
category	A string that contains the exercise category.
desc	A description of how to perform the exercise
date	Date that specifies when the exercise was added.
gbd_name	A string that contains the file name of the exercise
added_by_id	Foreign key, references the therapist id who added the exercise

Results

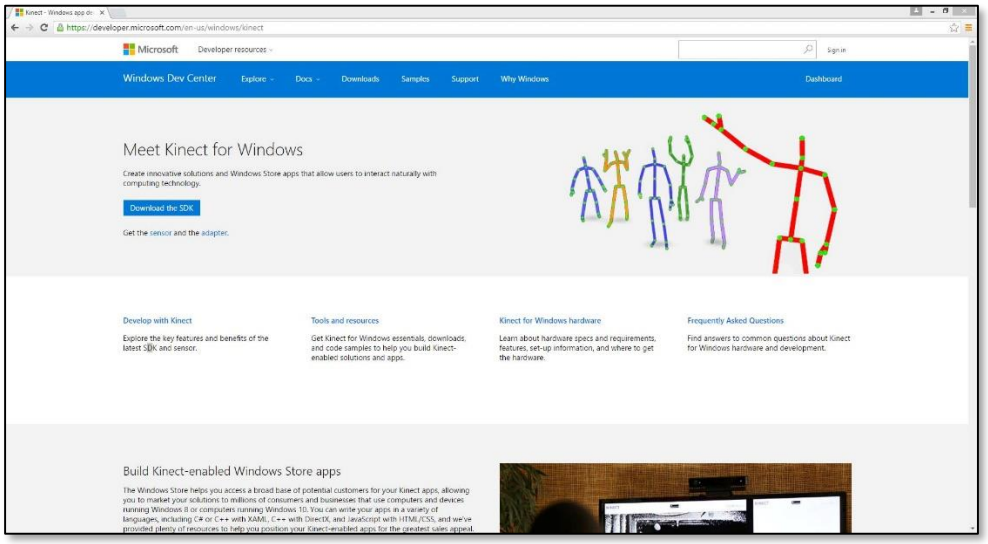
This table stores the results of session exercises.

Attribute Name	Attribute Description
id	Primary key, used to uniquely identify a given exercise in the database.
ordering	Field used to denote sequence of exercises in a session
date	Due date of the session the result belongs to
total_reps	Total number of reps to be completed by the patient
completed_reps	Number of reps completed by the patient
expected_time	Time allotted for the patient to complete the exericse
completed_time	Time taken to complete the exercise
patient_id	Foreign key, references the id of the patient who completed the
exercise_id	Foreign key, references the id in the exercise table

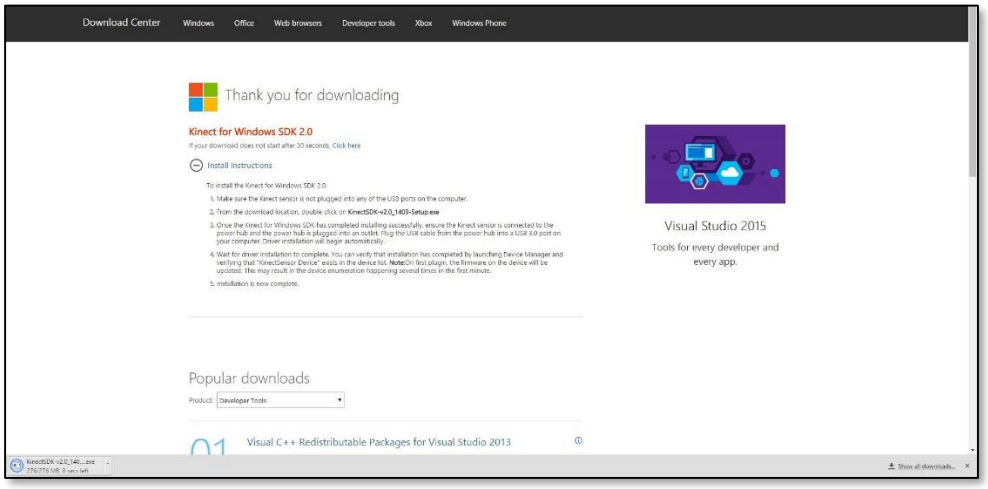
6 Microsoft SDK Tools

6.1 SDK Setup

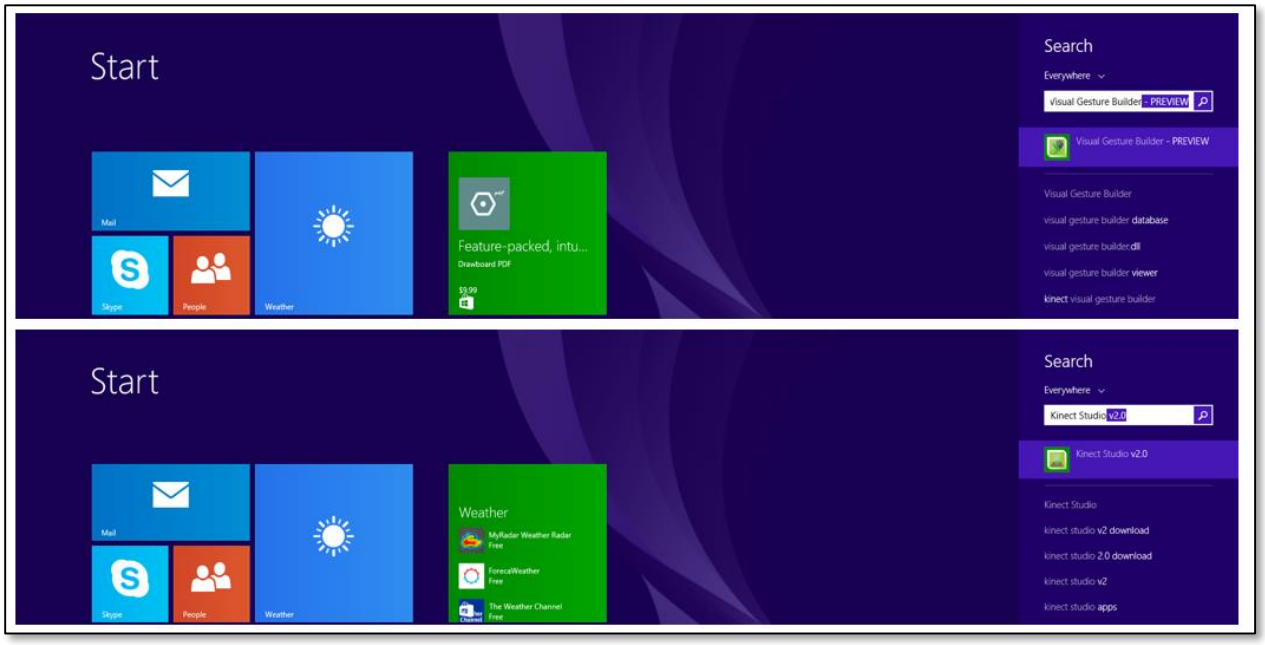
1. To acquire Microsoft Visual Gesture Builder and Kinect Studio download the Kinect v2 SDK from the following link: https://developer.microsoft.com/en-us/windows/kinect
The link should take you to a site that looks similar to the one below.



2. Click the "Download the SDK" button. A download should start. Run the executable and follow through the instructions.



3. Once you have downloaded the Kinect SDK, Visual Gesture Builder and Kinect Studio will be installed. The default download path: C:\Program Files\Microsoft SDKs\Kinect\v2.0_1409\Tools\KinectStudio. Alternatively you can search for Visual Gesture Builder or Kinect Studio with the Windows search tool.



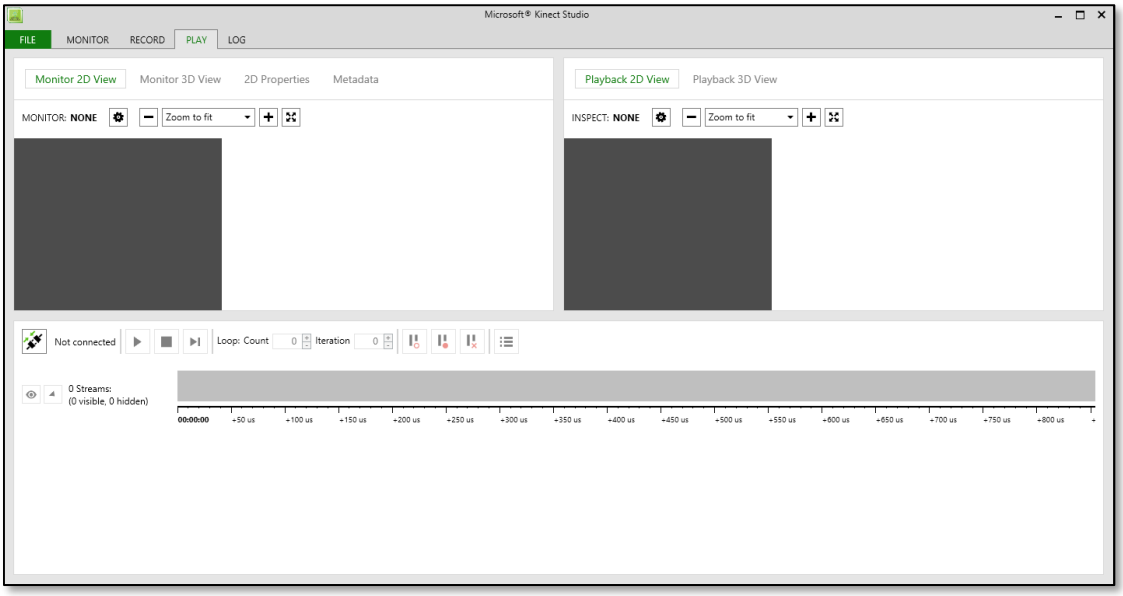
There is no additional setup to use these tools.

4. If you have not already at this point, plug the Kinect v2 into the Kinect for Windows Adapter and then plug the adapter into a power socket and a USB 3.0 or higher port in the PC. Wait for the Kinect Drivers to install automatically.

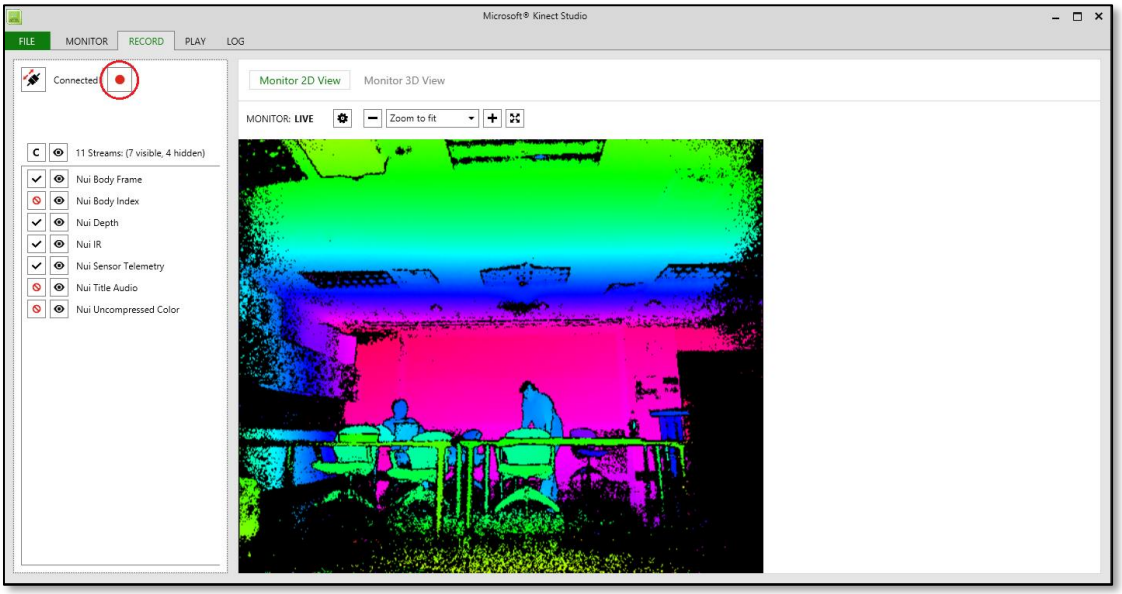
If you have any additional trouble up to this point you can check the official Microsoft forum at:
https://social.msdn.microsoft.com/Forums/en-US/home?forum=kinectv2sdk.

6.2 Creating New Exercises

1. To begin creating .gbd exercise files for upload to the website, launch Kinect Studio.



2. After ensuring that the Kinect is properly connected to the desktop, select the record tab and hit the record button. Then perform the desired exercise in front of the Kinect. This exercise should be performed by multiple people with differing body types and about fifteen repetitions per person for best results.

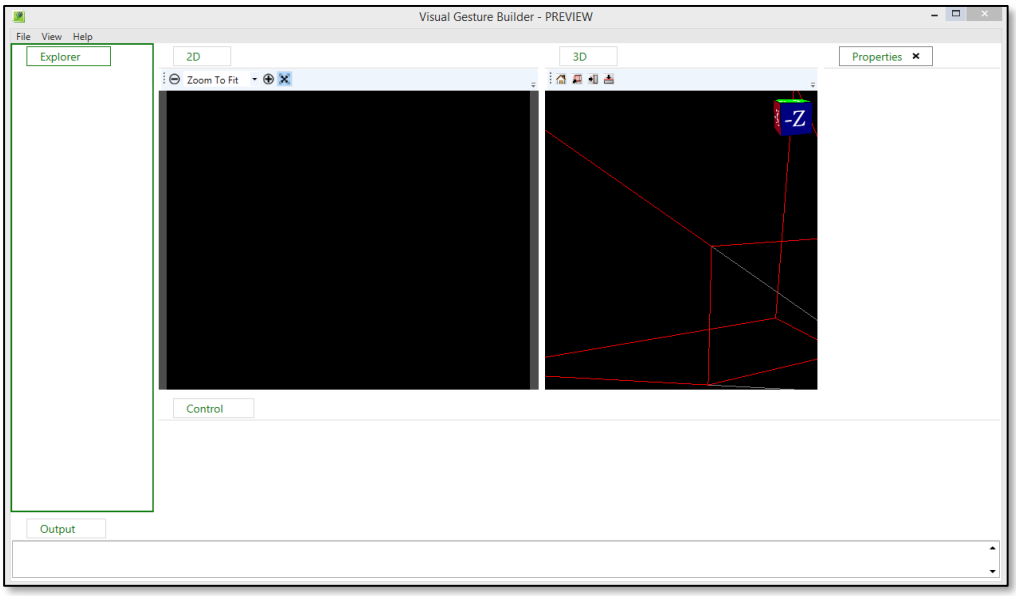


Once complete hit the recording button again and Kinect Studio will automatically save your file in:

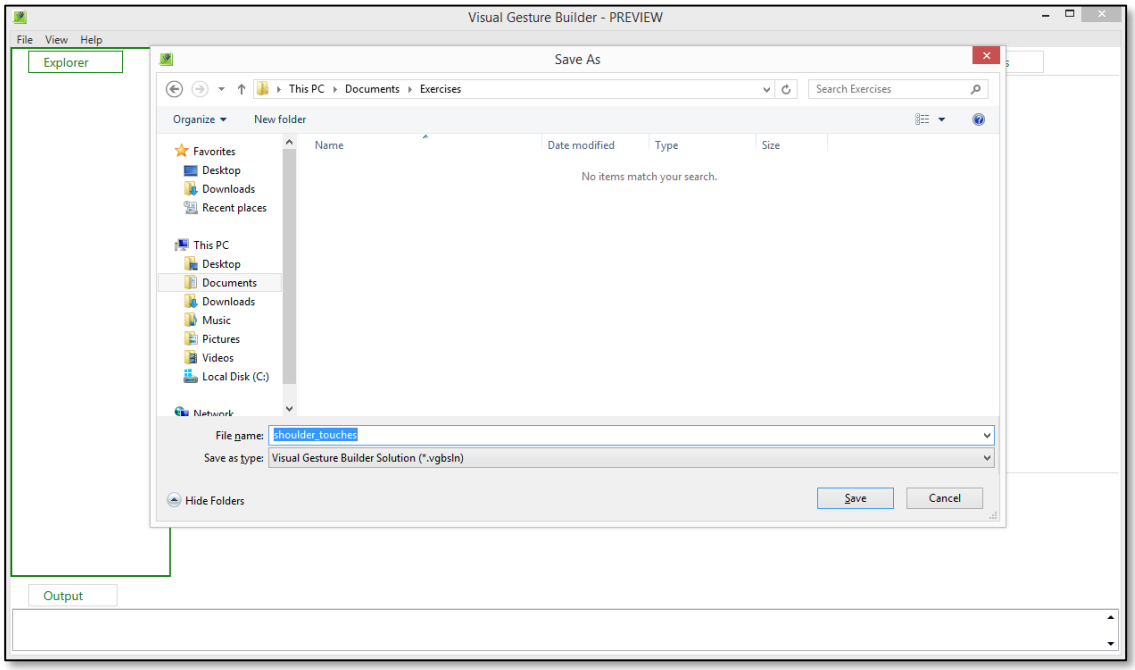
\...\Documents\Kinect Studio\Repository

It is recommended that your rename the generated video files so as to more easily keep track of which files to use in the next part.

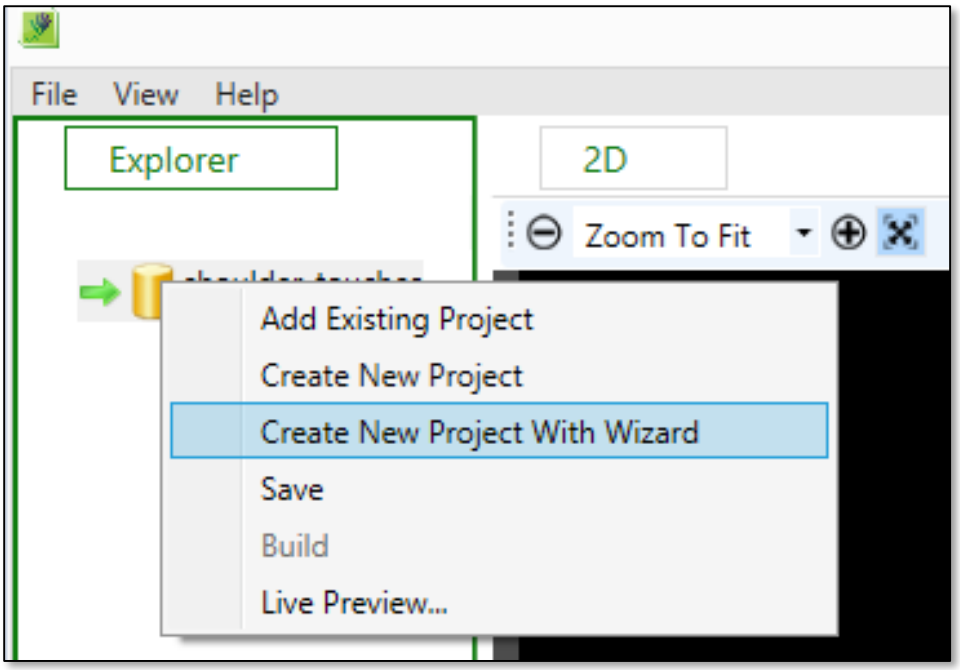
3. With the desired exercise recorded, Kinect Studio **must** be closed before you open Microsoft Visual Gesture Builder.



Now that you have opened Microsoft Visual Gesture Builder, select File->New Solution and save a new Visual Gesture Builder Solution.

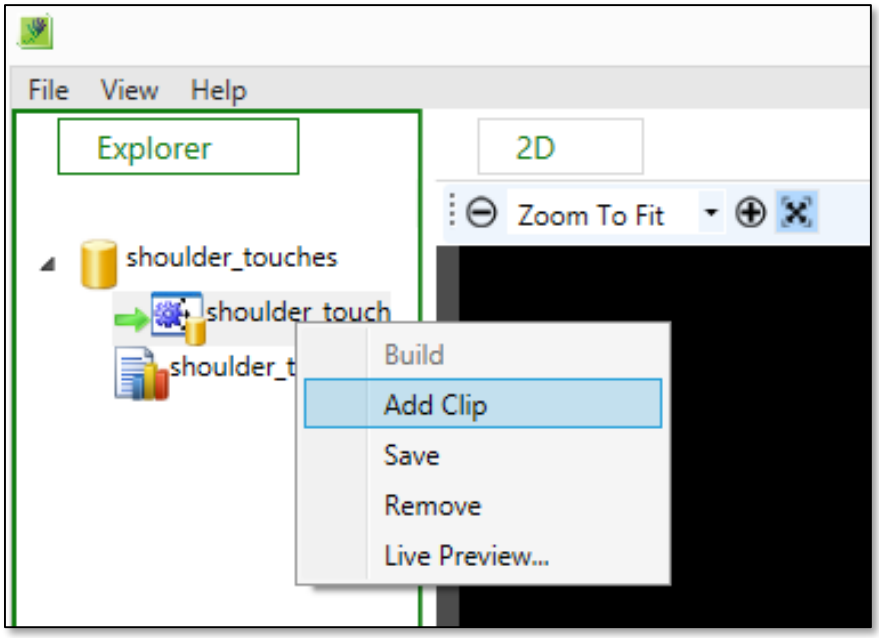


4. After you have saved the solution you'll be working on, right click on the solution and click Create New Project With Wizard.

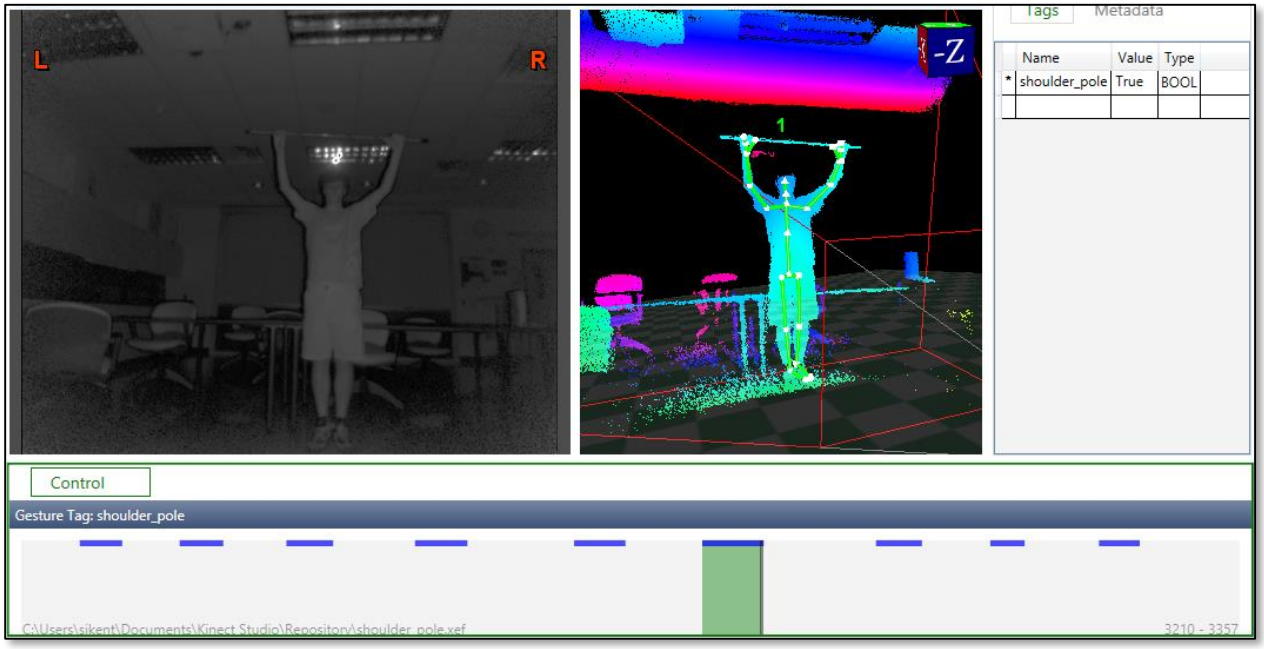


The wizard will then guide you through the steps of creating a new project.

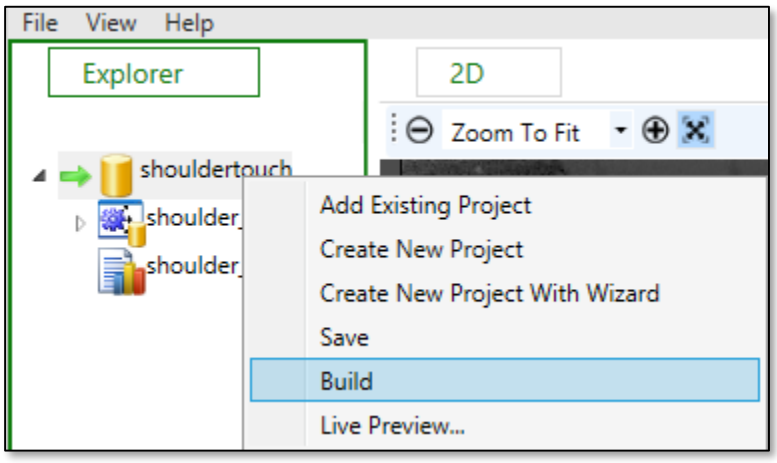
5. When you are finished setting up a new project, right click on the generated project file and upload the clip(s) that were created by Kinect Studio.



6. Navigation through the clips is performed via the arrow keys. To "tag" a gesture correct, the hold the shift key while navigating through the appropriate portions of the clip. Hit enter once the gesture is complete and repeat for all the gestures in the clip(s).



7. After you have finished tagging all of the appropriate portions of the clip(s), right click on the solution and hit "build". This will provide the .gbd file that will be uploaded to the website.



For a more in depth walkthrough, please view the following link: https://channel9.msdn.com/Blogs/k4wdev/Custom-Gestures-End-to-End-with-Kinect-and-Visual-Gesture-Builder

7 Glossary of Terms

Amazon Web Services – A secure cloud services platform for compute power, database storage and content delivery.

Amazon RDS – A scalable relational database in the AWS cloud

C# - A multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines

Django - Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

.gbd – Stands for Gesture Builder Database and is the file generated by Microsoft Visual Gesture Builder. It is responsible for gesture recognition

Kinect Studio – Kinect Studio is a utility application that you can use to preview Kinect sensor array data, record and play eXtended Event File (XEF) files, control the timeline position, and select 2D or 3D views. Kinect Studio APIs enable you to develop custom tools, to record and play back body data using XEF files.

Kinect v2 - a motion sensing input devices by Microsoft for PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures

Kinect for Windows SDK 2.0 – A set of developer tools, tutorials, and an API reference put out by Microsoft for the development of Kinect v2 on Windows 8, 8.1 and Windows 10.

Mono - Mono is an open source implementation of Microsoft's .NET Framework based on the ECMA standards for C# and the Common Language Runtime.

Monodevelop – A cross platform IDE for developing .NET applications on Linux, Windows and Mac OS X.

MySQL – A popular Open Source SQL database management system, developed, distributed, and supported by Oracle Corporation

Python - Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Nginx - A free, open-source, high-performance HTTP server and reverse proxy, as well as an IMAP/POP3 proxy server. It provides load-balancing, security controls and other monitoring tools.

.*NET* - A software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability.

Slack – A collaboration tool that allows for instant messaging, file sharing and other customizable plug-ins.

Telerehabilitation- the delivery of rehabilitation services over telecommunication networks and the internet.

Unity - a cross-platform game engine developed by Unity Technologies and used to develop video games for PC, consoles, mobile devices and websites.

USB 3.0 - The third major version of the Universal Serial Bus (USB) standard for interfacing computers and electronic devices. Among other improvements, USB 3.0 adds the new transfer rate referred to as *SuperSpeed USB* (SS) that can transfer data at up to 5 Gbit/s (625 MB/s), which is about ten times as fast as the USB 2.0 standard.

Visual Gesture Builder - Visual Gesture Builder (VGB) generates data that applications use to perform gesture detection at run time. By using a data-driven model, VGB shifts the emphasis from writing code to building gesture detection that is testable, repeatable, configurable, and database-driven.