Contents

Table of Figures

Table Index

1. ABSTRACT

‘Gesture Recognition for Education’ is a research project, under Project OSCAR at IIT Bombay.
The current system of education is not as interactive as it can be. It lacks participatory learning from the student point of view. The future of learning is visual and interactive. Keeping this in mind, the primary goal of this project is to interface 3D animation software and gesture recognition hardware to create an interactive learning environment. Blender is an Open Source 3D animation tool, which can be used for the modelling and animation aspects. Microsoft Kinect can be used for capturing real time gestures of users. In this paper we present a method for manipulating Blender models using Kinect. This Human-Computer Interaction method can be used to make various learning tools to make teaching more visual, animated and lively rather than mere reading from the theory.

The subject that we have targeted is Mathematics. So we developed MathMazing. It is a single player, gesture controlled mathematical maze. The player needs to guide the figure along the path leading to the correct answer. The figure moves along when relevant gestures are provided by the user through Kinect. This game will in turn improve the users Logic, Kinesthetic and Spatial skills. The target audience for this game is primary school kids. The final outcome is an application which can be distributed in an exe format to the schools who wish to use them. We believe that this form of education will support participatory learning.

2. LITERATURE SURVEYED

The traditional method of black board teaching, textbooks have been in place since long and are customary means of education. However, for certain concepts, actual practise is better.
The challenge is to make education intuitive from the user perspective while at the same time being useful, cost effective and feasible to implement at the grass root level. Using contemporary devices like laptops and LCD projectors in our education system is becoming common. In addition to this various e-learning sources like online videos, websites and e-books are also now freely available. Teachers are keen in using these in the classrooms.

2.1 Development Tools

Apart from the commonly used devices mentioned above there are new devices which are emerging. These include motion sensing devices, portable video recorders etc. So the development tools we selected include:

2.1.1 Hardware: Microsoft Kinect

It is a motion sensing input device by Microsoft for the Xbox 360 video game console. Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface using gestures and spoken commands [1]. The Kinect device is shown in Figure 1.

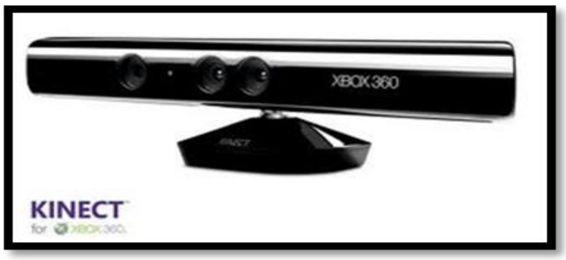


Figure 1: Kinect

2.1.1.1 Rationale for Selecting Kinect

- No controllers or remotes are needed.
- It supports 3D which is not supported by a normal camera.

- Kinect Effect is fascinating. Within the first 60 days, Kinect sold more than 8 million sensors, setting the Guinness Book World Record as the fastest-selling consumer electronics device.

2.1.1.2 Kinect Applications for Education

- Avatar Kinect: It lets user's avatar interact with up to seven other friends in 24 virtual stages ranging from a late-night talk show set to outer space to a sports tailgate party. To make the cyber socializing more real, Kinect's camera tracks gestures and facial expressions – eyebrow raises, smiles, belly laughs and all. [3]
- Synapse for Kinect: In this, body can be used as a Musical Instrument.
- CopyCat: An interesting approach to helping those who are hard of hearing or deaf is to have a kinect sensor recognize and offer feedback as a learning tool for sign language. CopyCat project incorporates educational games for small children in order to teach them sign language skills. [4]
- DaVinci for Kinect: Gestures are used to create objects and control the physics of the environment. [5]

2.1.2 Software: Blender

Blender is a free open source 3D graphics application, available under the GNU General Public License for the Linux, Mac OS X, Microsoft Windows operating systems. Blender can be used to create interactive 3D applications, games, animated film, or visual effects. [2]

Figure 2: Blender

2.1.2.1 Rationale for selecting Blender

- It can be used for modelling, animation, rendering and game logic.
- It is open source.

2.1.2.2 Blender Applications

- Big Buck Bunny: It is a short computer animated film by the Blender Institute. The plot follows a day of the life of Big Buck Bunny when he meets three bullying rodents. [6]
- Yo Frankie: It is an open source 3D platformer game, produced by the Blender Foundation. Frankie, an angry looking squirrel, is attacked by all animals in his surroundings and has to defend himself. [7]

3. PROBLEM DEFINITION

This project targets improving Mental Mathematical skills of the students. Other than the customary paper solving method of learning mathematics, several online tutorials and games have become popular. We intend to encourage students to practice mathematics daily by using gesture based input for our mathematical maze. These gestures make the game fun and interactive.

3.1 Scope

- To integrate Kinect and Blender.
- To demonstrate the Kinect and Blender integration through an educational application.
- To implement an educational application that uses gestures as input.

3.2 Objective

To record live gestures through Microsoft Kinect and map them to Blender models for making education more visual and simpler to understand.

4. PROPOSED SYSTEM

Since Blender and Kinect combination is currently being researched, there is no online support for Kinect in Blender as of now. One way is to use pre-recorded input from kinect as a .bvh file and map it to a skeleton in Blender. Its workflow is:

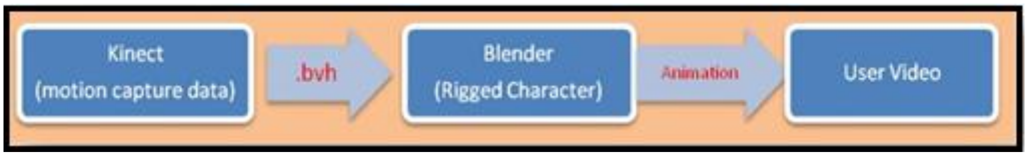


Figure 3: Flow Diagram

Technique 1:

To achieve the above the steps required [10] are as follows:

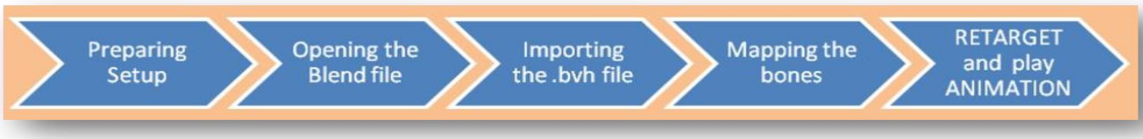


Figure 4: Technique 1 flow

Technique 2:

An alternative approach [11] is:

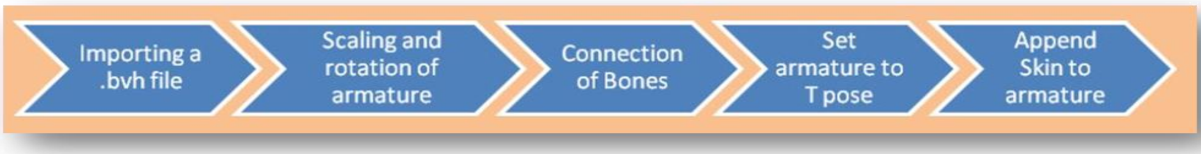


Figure 5: Technique 2 flow

4.1 Our approach to solve the problem

The major disadvantage of the above system is that it is offline. Hence, it is only useful for animations and not for live gesture recognition.

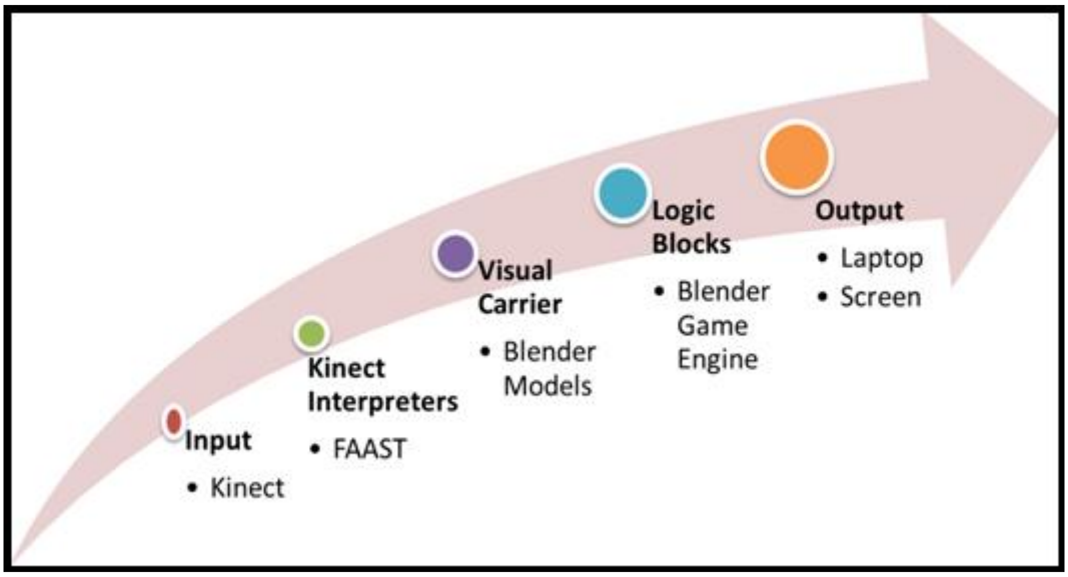


Figure 6: Our Approach

Since our aim is real time gesture recognition we will use a middleware to interface Kinect and Blender. The workflow is shown in Figure *6*. In this, Kinect acts as the input device. These inputs are given to FAAST (The Flexible Action and Articulated Skeleton Toolkit) which understands it and translates it in needed format. Its output is provided to Blender. All the modelling is done in Blender which acts as the visual carrier. The Blender Game engine runs the core logic of the game. Finally, the output can be projected on a laptop/screen.

5. PROJECT MANAGEMENT

Project management is the discipline of planning, organizing, securing, and managing resources to achieve specific goals.

5.1 Project Work Breakdown Structure

Table 1: Project Work Breakdown Structure

1	**Analysis Phase**
1.1	Study of existing system
1.2	Study and discussion of research done
1.3.1	Problem definition
1.3.2	Scope
1.3.3	Feasibility
1.4	Defining the problem
1.5	Fixing the scope of the project
1.6	Feasibility analysis
1.7	Requirement Analysis
1.8	Project Estimation
2	**Design Phase**
2.1	Developing the architecture of the system
2.2	Developing data flow diagrams of the system
3	**Coding**
3.1	Coding the logic of the mathematical maze
3.2	Coding FAAST instructions that interpret gestures.
4	**Testing Phase**
4.1	Unit Testing
4.2	Integration Testing
4.3	System Testing
5	**Deployment Phase**
6	**Documentation**

5.2 Schedule

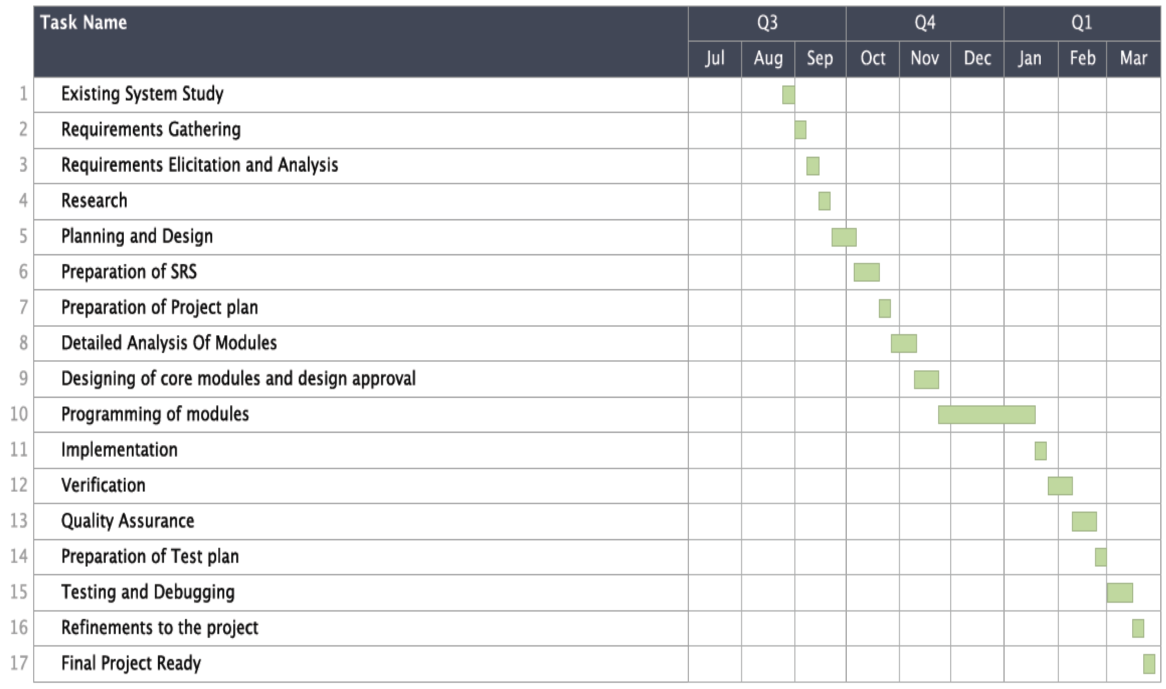


Figure 7: Gantt chart

5.3 Project Resources

The resources needed to run this project are:

5.3.1 Hardware

1. Microsoft Kinect
2. Laptop
3. Projector (Not Compulsory)
4. Screen (Not Compulsory)

5.3.2 Software:

1. 64 bit Windows Operating System
2. FAAST (Flexible Action and Articulated Skeleton Toolkit

5.4 Estimation

Estimation is basically identifying and acquiring necessary resources such as equipments, materials, man-power etc required for accomplishing the project successfully.

5.4.1 Project Estimates

The following are the estimation attributes to be calculated:

5.4.1.1 Lines of Code

Estimated lines of Code=600

The above mentioned Lines of Code (LOC) includes the code for:

- Interfacing Kinect with the system.
- Modules such as skeleton tracking, detection of gestures particularly hand.
- Processing of inputs from various above mentioned modules.
- Modules converting gestures into appropriate animation using logic.

5.4.1.2 COCOMO

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics.

COCOMO applies to three classes of software projects:

- **Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements
- **Semi-detached projects** - "medium" teams with “mixed” experience working with a “mix” of rigid and less than rigid requirements
- **Embedded projects** - developed within a set of "tight" constraints (hardware, software, operational)

Gesture Recognition for Education comes under the category of Embedded projects.

The Basic COCOMO equations take the form

Effort Applied (E) = ab(KLOC)bb **[man-months]**

Development Time (D) = cb(Effort Applied)db **[months]**

People required (P) = Effort Applied / Development Time **[count]**

Table 2: COCOMO Estimates

Software project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques and so on.

5.4.1.3 Estimate of Cost, Effort and Duration

According to the explanation given above, our project will be of type Embedded since it will be developed under time, software and hardware constraints.

Therefore we will take ab=3.6, bb=1.20, cb=2.5, db=0.32

Using the above values will calculate the following:-

Effort applied = ab(KLOC)bb

= 3.6*(0.6)^1.20

= 3

Development time = cb(Effort Applied)db

= 2.5*(3)^0.32

= 0.9

People required = Effort applied/Development time

= 3/0.9

= 3.33 ~3

Thus according to COCOMO model we get the following result

Table 3: Estimation Results

Effort applied	30 man-months
Development time	9 months
People required	3

6. PROJECT DESIGN

Project design includes an array of activities from generating ideas to planning how these ideas could become a realisable project. It basically represents a blueprint for a project.

6.1 System Architecture

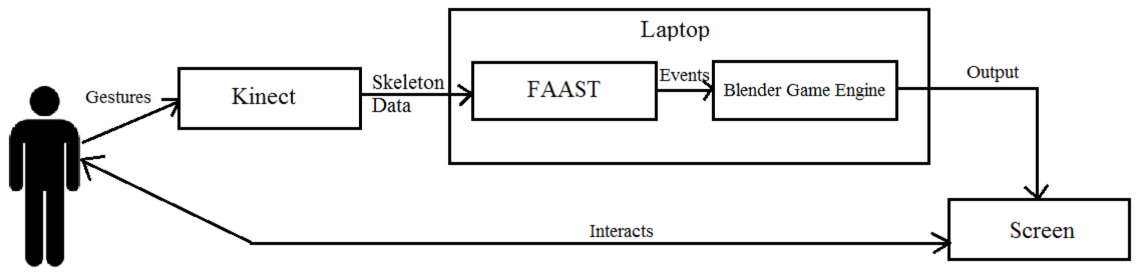


Figure 8: System Architecture

Details of the individual components are:

1. **Kinect:** It serves as the input device. It is a motion sensing input device by Microsoft. The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software", which provide full-body 3D motion capture, facial recognition and voice recognition capabilities. It will capture gestures of the user and pass it to FAAST. [1]
2. **Laptop:** The laptop will run FAAST and the final project executable prepared and processed by the Blender game engine.
   a. **FAAST**: It will help in interpreting the data captured by the device and passing it to Blender. The Flexible Action and Articulated Skeleton Toolkit (FAAST) is a middleware to facilitate integration of full-body control with games and VR applications using either OpenNI or the Microsoft Kinect for Windows skeleton tracking software. The toolkit can also emulate keyboard input triggered by body posture and specific gestures. This allows the user to add custom body-based control mechanisms to existing off-the-shelf games that do not provide official support for depth sensors. [8]

b. **Blender Game Engine:** It is a powerful high-level programming tool. It is responsible for imparting logic to the game. It uses a combination of "sensors", "controllers" and "actuators" to control the movement and display of objects in the engine. It also has its own Python API that enables python scripts to control the game. [9]

3. **Screen:** The final output can be projected on the screen to involve multiple users at the same time.

6.2 Modules

The various modules are:

1. Capturing User Gestures: The user’s gestures in real time need to be captured for live gesture recognition. These gestures will be captured by the Kinect device. It has a 3D depth sensor that captures the movements of the user in 3d space.
2. Recognizing and Interpreting Gestures: The system needs to recognize user’s gestures and then interpret the meaning of those gestures. When relevant gestures are performed by the user, the system should replicate corresponding actions in the system. We will be using FAAST as the middleware to perform this. FAAST will interpret Kinect inputs and trigger corresponding events in response.
3. Mapping Gestures to Player Actions: Based on the gesture the user performs in real time, corresponding actions are to be performed by the player in the virtual world. Thus user gestures are to be mapped to the player actions. We will be using the Blender game engine(BGE) to perform this. BGE will receive events sent by FAAST and based on the event manipulate the player in the virtual world.
4. Executing MathMazing Logic: MathMazing is a single player, gesture controlled mathematical maze. It consists of multiple levels, with each level having a mathematical quiz. Each level will have a randomly generated equation with multiple options. The player is represented by a 3D object in the maze. The figure moves along when relevant gestures are provided by the user through Kinect. The player needs to guide the figure along the path leading to the correct answer. If the player gets the wrong answer, he can replay the level and sharpen his skills. Once the player gets the correct answer, he has two options: either to replay the level or move to the next level. Complexity of the equations increases with higher levels. Each level is

timed so the player is not only expected to get the correct answer but to get it within the allotted time. Time available reduces with each level thereby training the kid to calculate faster. The logic for this game will be written in Python as the BGE has its own PythonAPI.

6.3 Data Flow Design

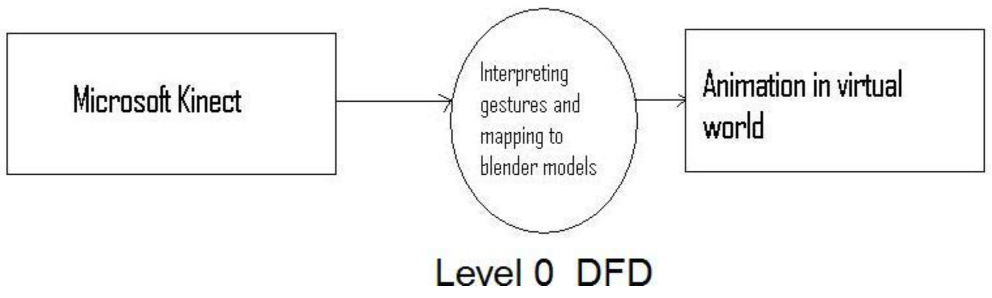

Microsoft Kinect
Interpreting gestures and mapping to blender models
Animation in virtual world
Level 0 DFD

Figure 9: DFD Level-1

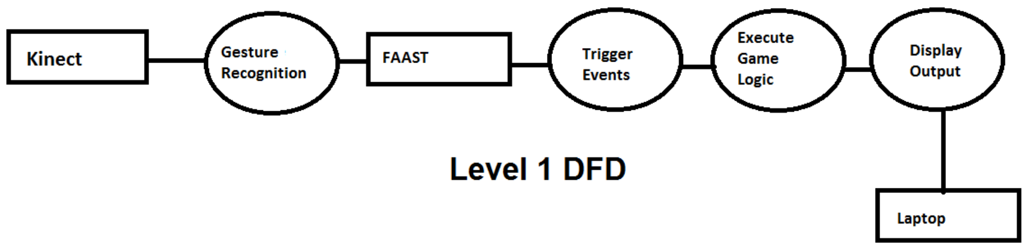


Figure 10: DFD Level-1

6.4 Interface Design

This describes design of various interfaces.

6.4.1 Hardware Interface

The system will require a Computer/Laptop and Microsoft Kinect.

6.4.2 Software Interface

Firstly, Kinect is designed for the XBOX console. So Computer/Laptop cannot understand the inputs from Kinect directly. Hence, drivers such as OpenNI, Nite Middleware and Prime

Sense Controller need to be installed. This will enable Kinect inputs to be understood by the Computer/Laptop.

As the Kinect inputs are interpreted by FAAST, the user must have it installed on his pc. Even if this software is not installed, it is freely available for download.

Mathmazing is an executable (exe) file. Therefore, Windows operating system is required to run the exe file.

6.4.3 External Interface

The user will interact with the virtual environment through gestures in front of Microsoft Kinect. These gestures will be replicated in the virtual world as output. To use Kinect for motion capture, a distance of 6 - 8 feet is to be maintained by the user from the Kinect device.

7. IMPLEMENTATION

The game titled as MathMazing integrates multiple intelligences. It is a single player, gesture controlled mathematical maze. It consists of multiple levels, with each level having a mathematical quiz. Each level will have a randomly generated equation with multiple options as shown below:

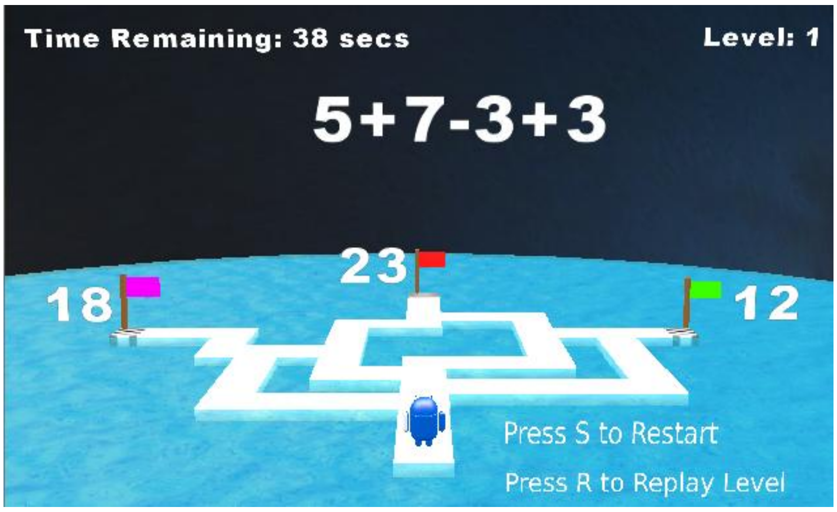


Figure 11: MathMaze

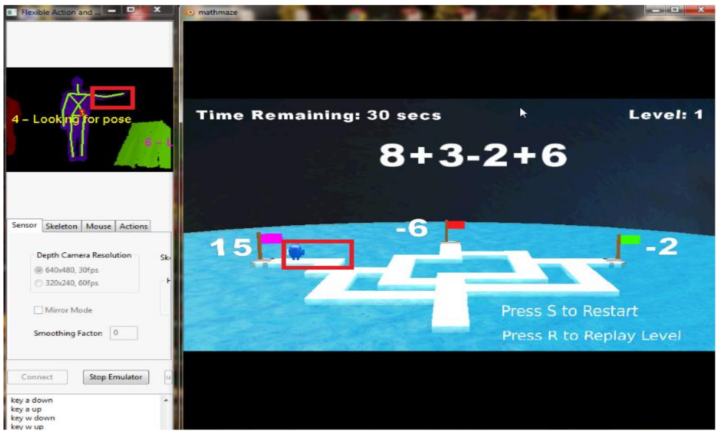
7.1 Development Process

The player is represented by a 3D object in the maze. The figure moves along when relevant gestures are provided by the user through Kinect. So first, we needed to capture user's gestures. For this, we used Kinect. Microsoft Kinect provides skeleton data in real time. However that data is understood by the XBOX Console and not the Computer/Laptop directly.

So we needed a software that could recognize and interpret Kinect data. After extensive research, we found FAAST, an open-source software that could interpret Kinect output and perform skeleton tracking in real time. Thus, we decided to use FAAST as a middleware between Kinect and Blender to support real time gesture recognition in Blender. In FAAST, we wrote a configuration file (.cfg) that provided instructions as to which gestures it should respond to and what event it should trigger when relevant gestures are provided.

```
# format: event_name threshold output_type event
```

Eg: left_arm_out 20 key_hold a



```
key a down
key a up
key w down
key w up
```

Figure 12: Level-1.a

In the game, the user needed to guide the player along the path leading to the correct answer. So, we needed to map the user gestures to player actions in the game. For this, we used the Blender Game Engine (BGE) logic blocks as shown:

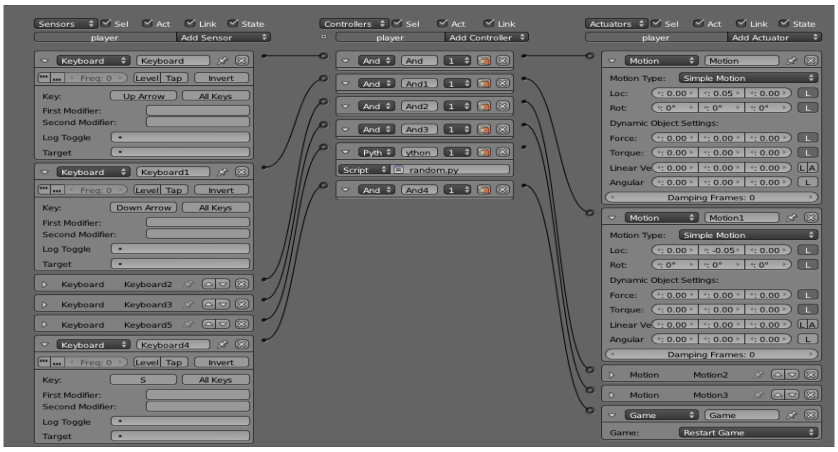


Figure 13: Blender Game Engine

Finally, we needed to implement the MathMazing game logic. This involved the following steps:

1. First we used static equation and options to test the game idea.
2. As static equation would make the game a one-time play, we needed to use dynamic equations. First, we thought of creating a database of questions and randomly selecting 1 question. On further brainstorming we realized that this would not be useful for daily practice as the questions would repeat. We needed completely dynamic equations of the same type. So, we kept the form of the equation constant and decided to use random numbers. For each level, we decided to use 3 operators

and 4 operands. Each of the 4 operands would be a random number from the sample space 1-9. In this way, our game can generate 9*9*9*9 (6561) different equations for each level. Thus, our game was completely dynamic.

3. We started coding Level 1. We decided to include only addition and subtraction in it. So we used the equation of the form:

   op + op – op + op //op is a random no between 1 to 9..

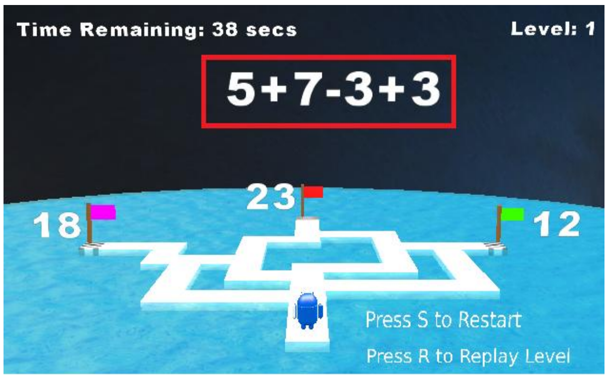


Figure 14: Level 1.b

4. We then needed to generate the 3 random options. Based on the equation form, we computed the maximum possible answer (25) and the minimum possible answer (-6). We then generated 3 random numbers in the range (min, max).
5. One of the options had to be correct and so we computed the answer (sum) of the equation dynamically generated. We then generated a random number in the range 0 to 2 (ans) and replaced the corresponding option with the correct ans.

   Eg: opt[0]=18 , opt[1]=23, opt[2]=6 // randomly generated nos

   sum = 12 // answer of the dynamic equation

   Ans = 2 // randomly generated no

   opt[0]=18 , opt[1]=12, opt[2]=12 // replacing option 2 with correct answer

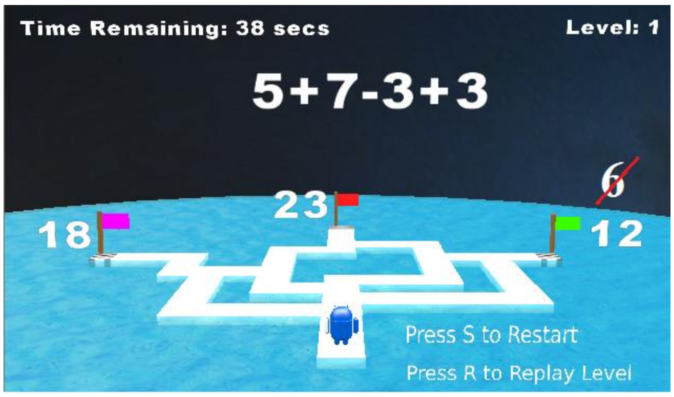


Figure 15: Level 1.c

6. We then needed to detect if the player reached the finishing line of the correct answer. We used logic bricks and Python code for this.
7. If the player gets the wrong answer, we gave him an option to replay the level and sharpen his skills.

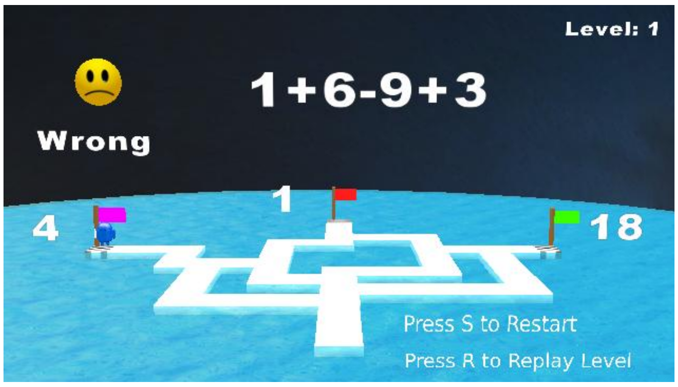


Figure 16: Level 1.d

8. Once the player gets the correct answer, we gave him two options: either to replay the level or move to the next level.

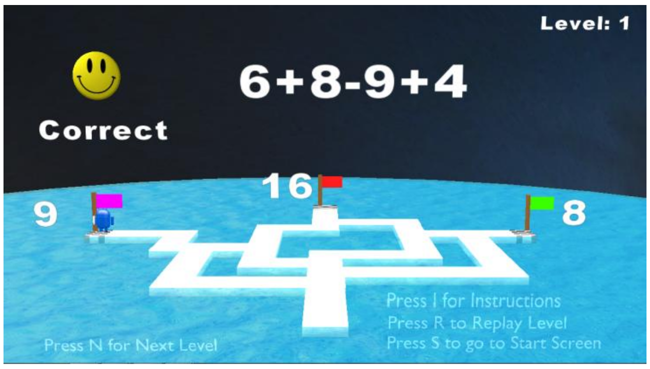


Figure 17: Level 1.e

9. We increased the complexity of the equations with each level. We introduced multiplication in level 2. So we used the equation of the form:

   op * op – op * op //op is a random no between 1 to 9..

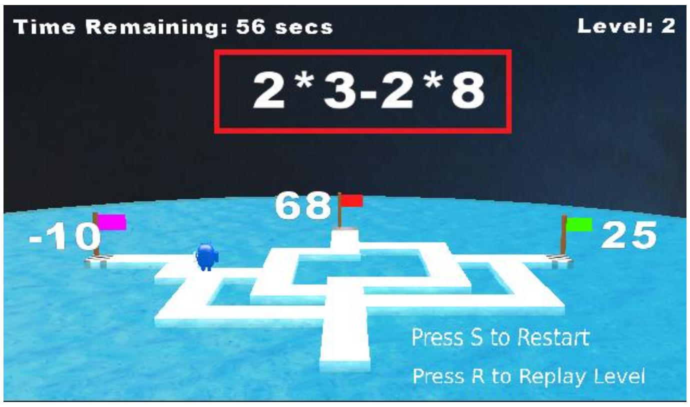


Figure 18: Level 2

10. We have used the BODMAS rule for Level 3. So we used the equation of the form:

    op – op + op * op //op is a random no between 1 to 9

11. Since, aim of the game is to sharpen children's mental math skills, we realized that we needed to have a timer for each level. Without the timer, the child would get infinite time to solve the equation, which would not be challenging enough. So the player is not only expected to get the correct answer but to get it within the allotted time.

12. We conducted informal interviews to determine how much time the kid's usually need for navigation and solving the different types of equations. Based on the results we used the following time limits:

    Level 1: 45 seconds since it is simple addition and subtraction

    Level 2: 75 seconds since it involves multiplication

    Level 3: 60 seconds since it is level 3 we expect the kids to calculate faster.

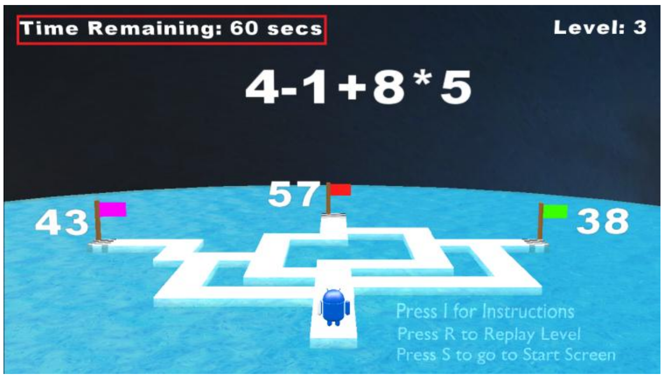


Figure 19: Level 3

8. PROJECT TESTING

Testing is the process which gives us a clear indication of how good, or rather, how bad has the system developed turned out. It is a good practice to start testing a system from its initial stages. We have followed the same procedure.

8.1 Test Cases

These are the examples of some of the test cases that were used to correct the errors in the system.

Table 4: Test Case ID-01

Purpose:	To check if a single user is tracked properly.
Unit to test:	Gesture Capture Device ie Kinect
Assumption:	Kinect captures the user properly
Steps to be executed:	Run FAAST User should stand within 6-8 feet from Kinect such that his entire body is visible. User should stand in T-Pose position to calibrate. Check if users calibration occurs and skeleton in being tracked properly
Actual Result:	The player is tracked properly.

Table 5: Test Case ID-02

Purpose:	To check if users gestures are interpreted properly.
Unit to test:	Kinect Interpreter i.e FAAST
Assumption:	The user is calibrated and being tracked.
Steps to be executed:	User should perform relevant gesture facing the Kinect. Check if FAAST triggers corresponding event in response to the gesture.
Actual Result:	FAAST interprets gestures correctly.

Table 6: Test Case ID-03

Purpose:	To check if the system properly tracks a single user even when multiple users are around him.
Unit to test:	Gesture Recognition Module
Assumption:	Gesture Recognition Module works accurately.
Steps to be executed:	Multiple users stand 6-8 feet away from the Kinect facing it. One of the users calibrates himself using T-pose. Once calibrated, check if system properly recognizes and responds only to his gestures.
Actual Result:	Even in the presence of multiple users, Kinect properly recognizes calibrated user’s gestures.

Table 7: Test Case ID-04

Purpose:	To check is users gestures are properly mapped to player’s actions.
Unit to test:	Gesture Mapping Module i.e Blender Game Engine Logic Bricks
Assumption:	The user is calibrated and being tracked properly.
Steps to be executed:	User should perform relevant gesture facing the Kinect. Check if the player performs corresponding action in the system.
Actual Result:	User’s gestures are properly mapped to player actions.

Table 8: Test Case ID-05

Purpose:	To check whether the Mathematical Maze runs properly.
Unit to test:	Blender Game Engine Logic Module.
Assumption:	User’s gestures are being properly mapped to player actions.
Steps to be executed:	Run the game.
Actual Result:	The Mathematical Maze runs properly.

8.2 Methods Used

We used the following methods for testing:

8.2.1 Unit Testing

We have considered each module as one unit and tested these units with help of test cases and test plan developed. Unit testing was carried out on each module and on every function within the module. It included testing the functions at extreme input values and for wrong input values.

8.2.2 Integration Testing

The modules of our architecture are integrated in order to see that they provide the necessary functionalities required. The various modules were tested together to check their accuracy and compatibility. For example, gesture recognition hardware was integrated with gesture interpreting software to check how one module takes input from other and works correctly.

8.2.3 System Testing

In this we have tested the system as whole with help of above test cases so that it provides proper output. This included regression and performance tests.

8.2.4 Validation Testing

In this we have given the system various inputs and tested if it provides proper and expected outputs. In case of any deviation from the expected output, corrective action was taken.

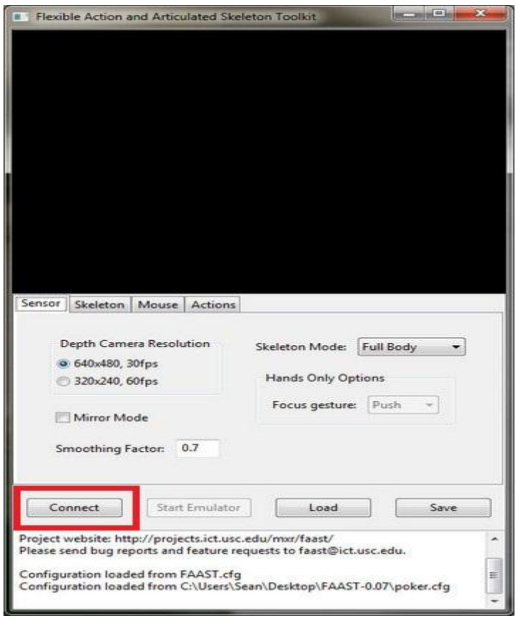
9. MAINTENENCE

Maintenance is basically concerned with the final deliverable of the project. It consists of User Manual and all the necessary help which can be useful for the user in the initial usage of the application.

9.1 User Manual

The following steps are to be followed to run the application successfully:

1. Connect Kinect with the laptop.
2. Start the FAAST application by double clicking on the icon.
3. Click on the Connect button (in the FAAST window) to connect laptop and Kinect.

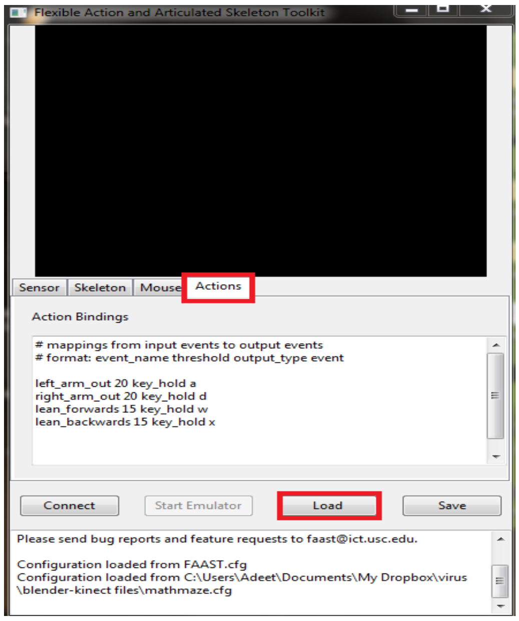


Depth Camera Resolution: [✓ 640x480, 30fps] 320x240, 60fps
Mirror Mode: ☐
Smoothing Factor: 0.7
Skeleton Mode: Full Body
Hands Only Options
Focus gesture: Push

```
Project website: http://projects.ict.usc.edu/mxr/faast/
Please send bug reports and feature requests to faast@ict.usc.edu.

Configuration loaded from FAAST.cfg
Configuration loaded from C:\Users\Sean\Desktop\FAAST-0.07\poker.cfg
```

Figure 20: FAAST-Connect

4. Go to the Action tab in FAAST and load the mathmaze.cfg file.



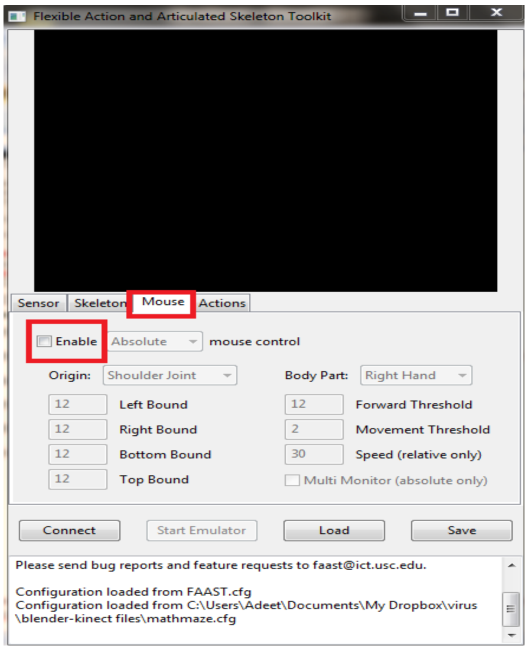
Action Bindings

```
# mappings from input events to output events
# format: event_name threshold output_type event

left_arm_out 20 key_hold a
right_arm_out 20 key_hold d
lean_forwards 15 key_hold w
lean_backwards 15 key_hold x
```

```
Please send bug reports and feature requests to faast@ict.usc.edu.

Configuration loaded from FAAST.cfg
Configuration loaded from C:\Users\Adeet\Documents\My Dropbox\virus
\blender-kinect files\mathmaze.cfg
```

Figure 21: FAAST-Action

5. Go to the Mouse tab and disable the mouse by clicking on the Enable checkbox. Now the Enable checkbox button should not be selected.



☐ Enable Absolute mouse control

Origin: Shoulder Joint Body Part: Right Hand

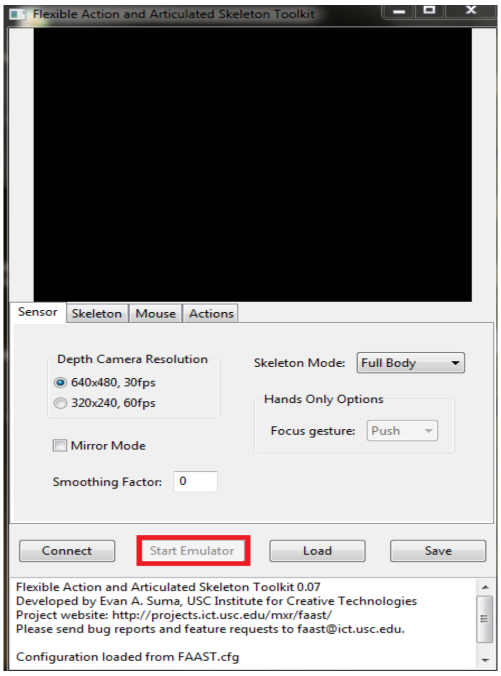
12	Left Bound	12	Forward Threshold
12	Right Bound	2	Movement Threshold
12	Bottom Bound	30	Speed (relative only)
12	Top Bound	☐	Multi Monitor (absolute only)

```
Please send bug reports and feature requests to faast@ict.usc.edu.

Configuration loaded from FAAST.cfg
Configuration loaded from C:\Users\Adeet\Documents\My Dropbox\virus
\blender-kinect files\mathmaze.cfg
```

Figure 22: FAAST-Mouse

6. Open the MathMaze executable (.exe) file by double clicking on the icon.

7. Press the Start Emulator button in FAAST.



Depth Camera Resolution
(●) 640x480, 30fps
( ) 320x240, 60fps

☐ Mirror Mode

Smoothing Factor: 0

Skeleton Mode: Full Body

Hands Only Options
Focus gesture: Push

```
Flexible Action and Articulated Skeleton Toolkit 0.07
Developed by Evan A. Suma, USC Institute for Creative Technologies
Project website: http://projects.ict.usc.edu/mxr/faast/
Please send bug reports and feature requests to faast@ict.usc.edu.

Configuration loaded from FAAST.cfg
```

Figure 23: FAAST-Start

8. Stand in front of the Kinect using the “T pose” as shown below:

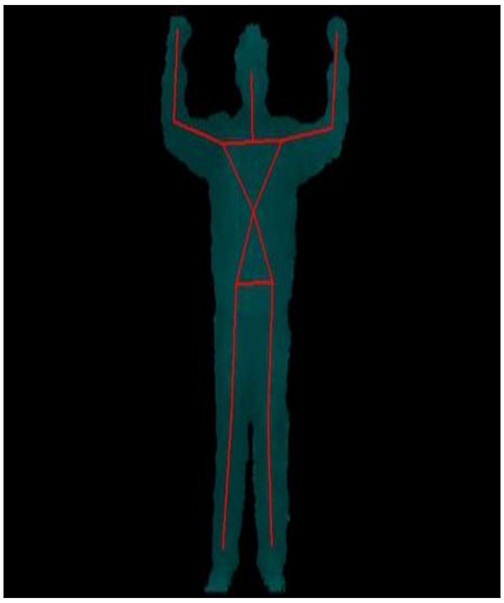


Figure 24: T-pose

9. As soon as you are calibrated (once the skeleton lines appear on your body in FAAST), you are ready to play the game.
10. Click on the Play button in the MathMaze.exe window.

11. Use the following gestures to move the player in the maze:

    i. Bend forward to move the player ahead.

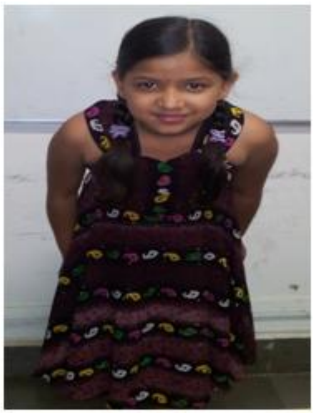


Figure 25: Forward

    ii. Bend backward to move the player behind.

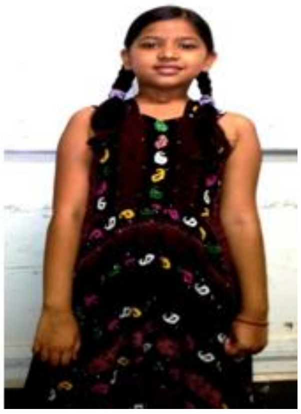


Figure 26: Behind

iii. Put your right hand up to move the player right.

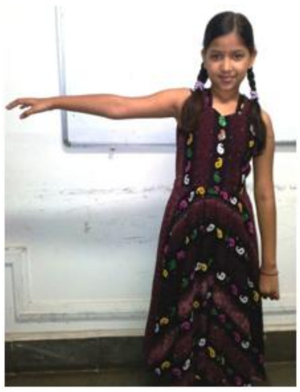


Figure 27: Right

iv. Put your left hand up to move the player left.

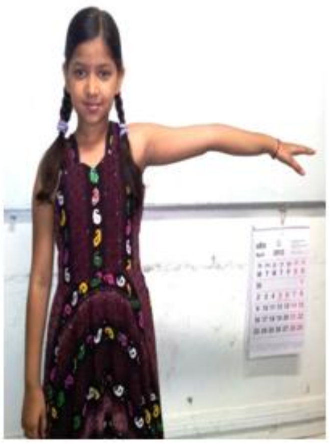


Figure 28: Left

To see the game demonstration, please view:
http://www.youtube.com/watch?v=aGja6TcBGFA

9.2 Constraints

1. Windows Operating System required as the .exe file would not run on any other OS.
2. The entire body should be visible to Kinect for proper tracking of the skeleton.
3. The maximum distance up to which Kinect can record gestures is 30 feet from Kinect but a distance of 6-7 feet is advisable for optimum working.
4. There should be less people around the user to avoid multiple tracking of skeleton.

10. CONCLUSIONS

This game improves the following skills:

- **Logical/Mathematical:** Solving of the equations to reach the correct answer improves children's calculation skills. It makes them faster and more accurate at basic Math.
- **Kinaesthetic:** The use of gestures increases the activity level of children and reduces boredom. It also improves players' stress levels, weight management, fitness, and health.
- **Visual/Spatial:** It develops the sense of location and direction for children. It improves their Spatial Awareness as they move objects through space.

The key features of this project are:

- **Open Source:** 'Gesture Recognition for Education' is an Open Source Project. The various tools used in this project such as Blender for modelling, FAAST for interpreting Kinect inputs are all Open source. As a result, this product will be freely available to all users
- **Intuitive:** Users can interact with the objects using natural gestures. Although it is a virtual experience it will be as good as a real one.
- **Simplifies Understanding:** User's gestures will manipulate objects in real time. This will enhance student's level of understanding as they will immediately see the consequences of their actions.
- **Learning with Fun:** Students love playing Kinect games. Thus by teaching them lessons while they play such games will increase their learning hours.

10.1 Future Scope

Using similar system architecture, we believe that it is possible to adapt it for creating more educational applications. The additional advantages of these applications would be:

- **Reduced Cost:** To perform an experiment in real world involves cost of the tools and apparatus. For instance, to perform a chemistry experiment each student will require separate chemicals. The involved chemical reaction will make the input chemical non-reusable. This increases the overall cost. On the other hand, performing the experiment in the virtual world involves no cost of tools, apparatus or chemicals as they will all be modelled using Blender which is free software.

- **Reduced Risk:** Several experiments especially chemistry experiments involve use of hazardous chemicals and explosives. Using such chemicals in real world will adversely affect student's health. However using such chemicals in virtual world will have no adverse effect in turn reducing risk.
- **Reusability:** Once the system is operational it can be used by several students. Every year different batches of students can use the same system thereby promoting reusability.

11. REFERENCES

[1] http://en.wikipedia.org/wiki/Kinect (last accessed on 23-02-2012)

[2] http://en.wikipedia.org/wiki/Blender_(software) (last accessed on 17-11-2011)

[3] http://www.microsoft.com/presspass/Features/2011/jul11/07-25KinectFunLabs.mspx (last accessed on 5-03-2012)

[4] http://kinect.dashhacks.com/kinect-news/2011/08/24/copycat-kinect-sign-language-learning-tool-presentation (last accessed on 12-01-2012)

[5] http://emergingexperiences.com/2010/12/davinci-goes-touchless-with-xbox-kinect/ (last accessed on 30-01-2012)

[6] http://en.wikipedia.org/wiki/Big_Buck_Bunny (last accessed on 13-10-2011)

[7] http://www.makeuseof.com/tag/yo-frankie-free-open-source-platform-game/ (last accessed on 03-02-2012)

[8] http://projects.ict.usc.edu/mxr/faast/ (last accessed on 21-03-2012)

[9] http://wiki.blender.org/index.php/Doc:2.4/Manual/Game_Engine (last accessed on 12-03-2012)

[10] http://www.vimeo.com/26420002/ (last accessed on 1-12-2011)

[11] http://www.youtube.com/watch?v=rxllmUsS_v0/ (last accessed on 17-11-2011)

[12] http://centralsource.com/blender/bvh/files.htm/ (last accessed on 19-10-2011)