Virtual Skiing

Tomazic, Tomaz; Lai Chin Chung, Louis

University of Ljubljana, Faculty of informatics and computer science

Abstract

This paper describes a virtual skiing game using kinect and unity 3D.

Keywords: skiing, kincect, unity 3D, virtual

1. Introduction

The interactive installation "Virtual skiing" enables a visual immersion into the feelings of gliding on snow through a winter landscape. The computer rendered winter landscape is displayed over the entire wall in front of the skier. As on real skis you can regulate the speed of descent by changing the posture of your body so that the air resistance is decreased or increased. By shifting the weight of your body to the right or left ski you can make turns down the slope between the snow capped trees. The interface to the virtual world is implemented by computer vision techniques which capture the posture of the skier's body. [1]).

2. Software overview

2.1. Goal of the project

The goal of the project is to provide the user a virtual skiing experience. To do so, at least the following needs to be done:

(i) track body movement of the user

(ii) display a virtual 3D environment on a screen

(iii) play game in this enviroment

2.2. Kinect

Overview. Kinect is the most convienent body movement tracking hardware we can find in the market, it thus becomes the tool for our project. The Kinect sensor haveo one RGB camera and two Infra-red sensors, allowing it to track movement to/from any direction in a 3D space.

Code. By examining the sample code, a skeleton of user is made using C#. A skeleton consist of nodes and lines, nodes representing the the joints of the user, lines representing the limbs/body of the user. By comparing the positions nodes, we can capture the user's movement and turn them into inputs in the game.There are 2 main types of input in the skiing game, one controls the direction and the other controls the speed of skiing.

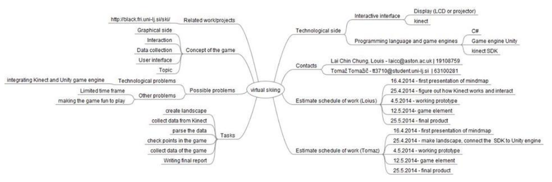
virtual skiing
Related work/projects
http://black.fri.uni-lj.si/ski/
Concept of the game
Graphical side
Interaction
Data collection
User interface
Topic
Possible problems
Technological problems
integrating Kinect and Unity game engine
Other problems
Limited time frame
making the game fun to play
Tasks
create landscape
collect data from Kinect
parse the data
check points in the game
collect data of the game
Writing final report
Technological side
Interactive interface
Display (LCD or projector)
kinect
Programming language and game engines
C#
Game engine Unity
kinect SDK
Contacts
Lai Chin Chung, Louis - laicc@aston.ac.uk | 19108759
Tomaž Tomažič - tt3710@student.uni-lj.si | 63100281
Estimate schedule of work (Loius)
16.4.2014 - first presentation of mindmap
25.4.2014 - figure out how Kinect works and interact
4.5.2014 - working prototype
12.5.2014- game element
25.5.2014 - final product
Estimate schedule of work (Tomaz)
16.4.2014 - first presentation of mindmap
25.4.2014 - make landscape, connect the SDK to Unity engine
4.5.2014 - working prototype
12.5.2014- game element
25.5.2014 - final product

Figure 1. Mindmap

Directions. To control the direction of ski, user need to band their body left or right to turn, or stay in the middle to go straight. A simple method is used to extract this particular movement by the user, we simply compare the nodes on both shoulders. The data obtained is then threasholded into only 3 inputs, left, middle and right.

Speed. The user need to squat to increase the speed, and detection of squatting is needed. When squatting, the position of the node on hip is changing obviously. So, the node on hip is used to determine whether the user is squatting or not.

3. Developing process

3.1. Planning

Mindmap. A mindmap is made to record all the ideas about the project. It gives the team some guide and direction to work on. This is also the step where we know each team member's strength and weakness, leads to better allocation of work later on.

3.2. Research

At the beginning of the project, none of the team member knows anything about programming on Kinect, it is the main focus of search. We found that Kinect SDK is essential in coding the sensor, it comes

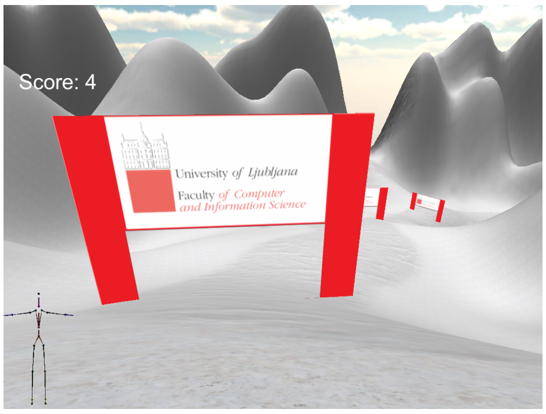

Figure 2. Screenshot form skiing game.

together with Kinect studio and a developer toolkit. Kinect studio is just for testing and the developer toolkit is a package of code examples.

For the displaying of the virtual skiing world, it took us almost no time to search for game engine, one of the team member knows how to use Unity 3D, certainly this becomes your choice of game engine.

3.3. Divide the work

The work of the project can be divided into two main parts, the input(Kinect) and output(the display). Chin Chung will work on Kinect, figure out how it works; Tomaz will work on the game engine, create a virtual 3D world for the user to ski on.

3.4. Unity

Unity is a development engine for the creation of 2D and 3D games and interactive content. It has multiplatform support and own asset store where you can download scripts, objects, scenes.. [2] We found kinect wrapper package[3] for Unity3D engine, which added us easy option to get collected data from kinect directly to our script code in unity. We used this data to calculate angle between different joints. With this angle we are then able to set intensity of turning sensitivity. To gamify our enviroment we put score counter in top left corner, which get increased by colliding carracter with advertised flags on the way down. To allow kinect detecting the human body and give user time to prepare we put countdown at the beginning of game.

Terrain. It has great support for creating terrain. In game we created simple terrain with mountains and put snow on it. For more flexible terrain we could create scirpt or just few images of pre-difined heightmaps. Unity have option to generate terrain using hightmap.

Scripting. Unity support C#, javascript and Boo scripting languages. It also supports applying different languaged script on same object. We used such ability for user movement in skiing game, using javascript and C# languages only.

3.5. Kinect

The first attemt to work on Kinect was a failure, not until I start working on it, I didn't notice the underlying problems. I did get the skeleton of the user, retrieving some data from it, but they are not very useful. I faced three main problems in this part.

C#. I did learn several languages before, but not C#. When I read the code, I understand them in a very low level, I recognize almost all the syntex, the loops, conditionals, declarations... etc. It took me really a lot of time to get to a higher level of understanding. The documentations from Microsoft was not very useful, it maybe enough for an experienced C# programmer, not for me.

Mathematics. fter a lot of reading, I finally locate the part where the orientation of the nodes/joints are stored. The orientations of the nodes are expressed in rotation quaternions , an absolutely alien term for me. I checked the documentation, wikipedia and other sites, they all explain it is a very mathematical way. The main direction I studied is software engineering, the mathematics they are using are far beyond my level. I just leave the output in the form of rotation quaternion.

Developing evniroment. I used the Microsoft visual studio in the part, mainly because most tutorials on Kinect are also on this environment. Problem comes when my part needs to merge with the game engine, I have no idea how to do it. Just importing the scripts in visual studio was not enough, the scripts will not run automatically. Scripts need to link to game objects in order to get running. Also the displaying machanism is very different in Visual Studio and Unity 3D, I don't know how to translate the code.

3.6. Rework

A replan is needed because the Kinect part and the Unity 3D part of the project was not mergeable. Another research is done to find Kinect code example directly in Unity 3D, not in Visual Studio.

Kinect in Unity 3D. Code example was found in Unity 3D wiki, this time the whole Kinect part is coded only in the game engine. Work done in Visual Studio is repeated in Unity 3D, it was a lot smoother than before because of the experience gained. Instead of the rotational quaternion, a simpular axis system is used to compare the postions of the Joints.

C# and Javascript. The code is until not fully compatable, the graphics part uses Javascript and the Kinect part uses C#. A way to transfer data between scripts in two langauges is found, and the problem is fully solved.

4. Conclusions

We were able to make game playable by setting correct parameters such as speed, turning speed, gliding and gravity accelleration. Project has been succesfully finished and requirements has been fullfield.

References

[1] Solina Franc and Batagelj Borut (2005) *http://black.fri.uni-lj.si/ski/*

[2] Unity 3D (2014) http://docs.unity3d.com/ScriptReference/index.html

[3] Microsoft SDK http://wiki.etc.cmu.edu/unity3d/index.php/Microsoft_Kinect_-_Microsoft_SDK