Bachelor Thesis

CREATIVE DANCE LEARNING PLATFORM USING MICROSOFT AZURE KINECT

Panagiotis Melios

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

MAY 2021

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Creative Dance Learning Platform Using Microsoft Azure Kinect

Panagiotis Melios

Advisor

Andreas Aristidou

Diploma project has been submitted for partial fulfillment of the requirements of Informatics Degree acquisition from the University of Cyprus

May 2021

Acknowledgement

First, I would like to express my sincere gratitude to my advisor Assistant Professor Andreas Aristidou, for his consistent support and guidance during the running of this project. During the year he push me towards the best results. However, he provided me with a PC so I could work on my bachelor thesis. Additionally, I would like to give my gratitude to Andreas Andreou (Qiiwi Games) for providing assistant where needed.
Finally, I would like to thank my family and friends for their support and patience.

Abstract

Dancing is a way a person can exercise and reduce his stress. This platform will help users learn Greek-Cypriot folk dances at home, with an animated tutor.

Firstly, user will have the chance to log in or register an account. Afterwards, user can choose a specific dance to learn. From there, the system will begin capturing all the dance movements of the user and will compare them with the animated tutors. A proper feedback will be shown throughout the lesson, so the user will be able to know if a movement, or a turn was done wrong or correct.

The user will have the option to choose either to play the whole dance or to play parts of the dance, so that he can learn or improve subsets of a dance, such as turns, specific movements, or the tempo of the dance.

The comparison of user's movement and tutor's is happening every second of the game. In more detail, the comparison is happened only for the translation of each bone of the user. Rotation and of course Scale are not included in the calculations.

This platform was developed in Unity 3D Game Engine. The reason I chose to develop this platform in Unity is that Unity is free. Also is a user-friendly application, thus is much more convenient and easier to use and of course it has a supportive community. The most important reason that I chose Unity is the Unity Asset Store, where I used a few assets that I acquired for free. Apart from free assets, Unity offers many useful packages, such as Text-Mesh-Pro, Cinemachine etc. Although, it is very easy to find Online tutorials, so it makes it even easier to learn and understand Unity.

In this thesis, I will show how I managed to achieve this comparison while using these technologies, inside a game of dance. The source code of this project is stored in the Black Window PC in Graphics Lab room 123.

Contents

Chapter 1

Introduction

1.1 Motivation

The goal of this platform is to give the chance to users to learn or improve Greek-Cypriot folk dances, in a user friendly and realistic environment. Dance is a type of entertainment that almost everyone loves. Dance has many advantages, including the relief of discomfort, the ability to keep in shape, and the improvement of one's figure. Nowadays, many people that are willing to take dance lessons, cannot achieve that, because of their busy working schedule. This platform will aim to satisfy these categories of users, not only those who work late, but also those who are shy and prefer dancing alone at home. Of course, in case of a pandemic, like COVID-19, is the safest way to entertain yourself while been confined at home. And, instead of going to a dance school twice a week, the user has the opportunity to engage with this application nearly every day. The performance of the user will improve as a result of practicing much more in gaming than in dance school.

1.2 Challenges

However, achieving this kind of platform has some challenges. Capturing the movement of the user in real time and comparing it with the specific level of dance. Although, capturing a dance turn and in general providing a full body capture was very challenging, that's why Microsoft Azure Kinect SDK[1] was used, specifically Azure Kinect Body Tracking SDK with its features. Body Tracking SDK provides body segmentation, contains an anatomically correct skeleton for each partial or full body in FOV[2], offers a unique identity for each body and can track bodies over time. Finally, another challenge was to figure out how to give the

[1] Software Development Kit
[2] Field of View, the extent of the observable world that is seen at any given moment.

user a rhythm lesson if he had no prior experience or understanding of the dance. Therefore, user has the option to learn a dance where the dance is divided into parts and letting the user understand each part first.

Synchronizing music with dance will also be a challenge, as it is something that it cannot be done by code. Synchronizing a dance with a music can only achieved manually. Hopefully, a lot of programs have the resources to let a developer achieve this synchronization.

1.3 Objectives

This platform’s aim is to use serious gaming to assist users in learning a dance and its rituals in a simple and quick manner. The setting is an old mansion, as well as the models are wooden dancers dressed in the traditional clothing of a Cypriot male dancer. The user will be able to practice a dance on his own while watching a mentor demonstrate the dance in real time, providing a more meaningful experience for the user.

Another goal was to associate Kinect with Unity, in addition to having a user-friendly and practical game. Gather data from the Kinect, skeleton mapping, and, in particular, the translation of each bone of the user.

The user for each dance will have two options either to play the whole dance or learn a dance. In this case the tutor will be played in parts, and the evaluation of user's performance will be happened for every part. This is another objective of this platform, to divide the animation into parts. If a user fails a part with a 50 percent or below average score, the part must be repeated. As a result, by failing and repeating a segment, the user can gain a greater understanding of the dance he picked. Therefore, we want to create a platform to be as realistic as possible, similar to a dance school, with the accompanying feedback message, for the user’s performance.

Chapter 2

Literature Review

2.1 Motion Analysis and Comparison

Motion analysis is based on the principles of Laban Movement Analysis LMA has four components which are Body, Effort, Shape and Space and it is used in A. Aristidou et al. [3]. This algorithm captures these four LMA components and uses them to achieve comparison and evaluation. This algorithm was used in a dancing platform where the student observes the virtual 3D teacher performing dance movements and repeating them while the user being monitored by a motion capture system.

A similar work that uses LMA components, but apart from LMA there are a few more techniques that have been used such as Music-related Motion Feature and VR (Virtual Reality) environment [4]. The VR environment is very helpful as it provides a mechanical interaction between the user and the virtual character. In this work the user is able to see the teacher in a VR mode and dancing with the teacher in real time. Although, this work is based on Guidance, Rhythm and Style. The final score of the user is based on the number of successful guidance attempts compare to a reference number and provided at the end of the session as reward.

A motion analysis can be achieved by comparing the motion with another motion [5]. This comparison was implemented in this work while using a Kinect-based Skeleton Tracking. This work describes a 3D environment that automatically evaluates dance performances against a standard performance and provides virtual feedback to the performer. The system uses an old version of Kinect to acquire the motion of the user via Kinect-based human skeleton tracking. The Evaluation of the Dancer in this system is calculated from three scores: Joint Positions, Joint Velocities, and 3D Flow Error.

2.2 Dance digitization and analysis

There are many dance commercial games, that can be played either with Kinect or with a combination of Kinect and Virtual Reality technology. These games have some differentials on feedback, for example the feedback of **Dance Central** is just contour blushes if any part of the body is wrong. **Just Dance** just shows the score and a feedback text, that is either equal to OK, Good or Perfect based on the accuracy of the user. However, the comparison of this game is happening with only on figure each time. **Dance Paradise** has a similar comparison with Just Dance, figure by figure comparison, as well as it is not a continuous dance.

In our case, the platform will evaluate the user’s performance through a continuous dance. Dancing a whole dance for the first time is very difficult especially for amateur users, therefore the system will provide two options for each dance, the first option is "Play Dance", and the second one the "Learn Dance". The difference of these two options is that the first option will play the whole dance and the second one will play the whole dance divided into parts. In both parts the user will receive a real time feedback and a final score.

2.3 Comparison with other works

The preview works are very similar to each other, as they use principles based on the LMA, to achieve comparison and evaluation. The comparison and the evaluation of the movement in this work will be based in one feature of the LMA. This platform will use the Shape component, as well as the joint transformation, hands and legs magnitude, and finally the speed of each joint.

The idea of this platform is that the student will observe the virtual 3D tutor performing dance movements and repeating them while being monitored by a motion captured system. The idea is similar to the concept of the *Folk Dance Evaluation* work [3]. In this platform the motion capture system, will be the Microsoft Azure Kinect. Kinect will be responsible for the motion capture of the user. With this system, the platform will be able to capture turns and spins of the student accurately, as in previews works spins and turns was very difficult to capture.

For the gamification part, the platform will give feedback to the user in the form of score for each movement, in each second. The feedback will contain the following messages:

“Perfect”, “Good”, “Bad”. The final score will be based on the average successful dance movements of the user compared to the 3D tutor. In other words, the comparison will be figure to figure for continuous dance, instead of being one figure each time. A difference from other games apart the comparison, is that this platform will provide a learning mode, where the animation of the dance will be divided into parts.

Chapter 3

Data Acquisition and Tools

3.1 Teacher’s Movement

The teacher movements are pre-captured templates. Professional dancers have been captured in Computer Science Graphics Lab in University of Cyprus, using a high-quality motion capture system. All the dances that were captured are in an online database of the university. This database stores a large variety of Cypriot folk dancing, as well as contemporary and Latin dances. The dances are available as fbx, bvh and C3D to download as well as the mp4 video of the recorded dancer. Because, the development of this platform will be in Unity, the best option was to use the fbx extension of the dances, as it will be easier to change some parts of it, these parts are needed for the training option.

All the available dances of the platform, where first opened as motion files in MotionBuilder, where they were merged with a characterized Cypriot traditional dancer. The synchronization of music and dance became increasingly important after that. MotionBuilder was again used to do this. This part, of how the motion file was merged with the traditional character is explained in more details in Chapter 6.

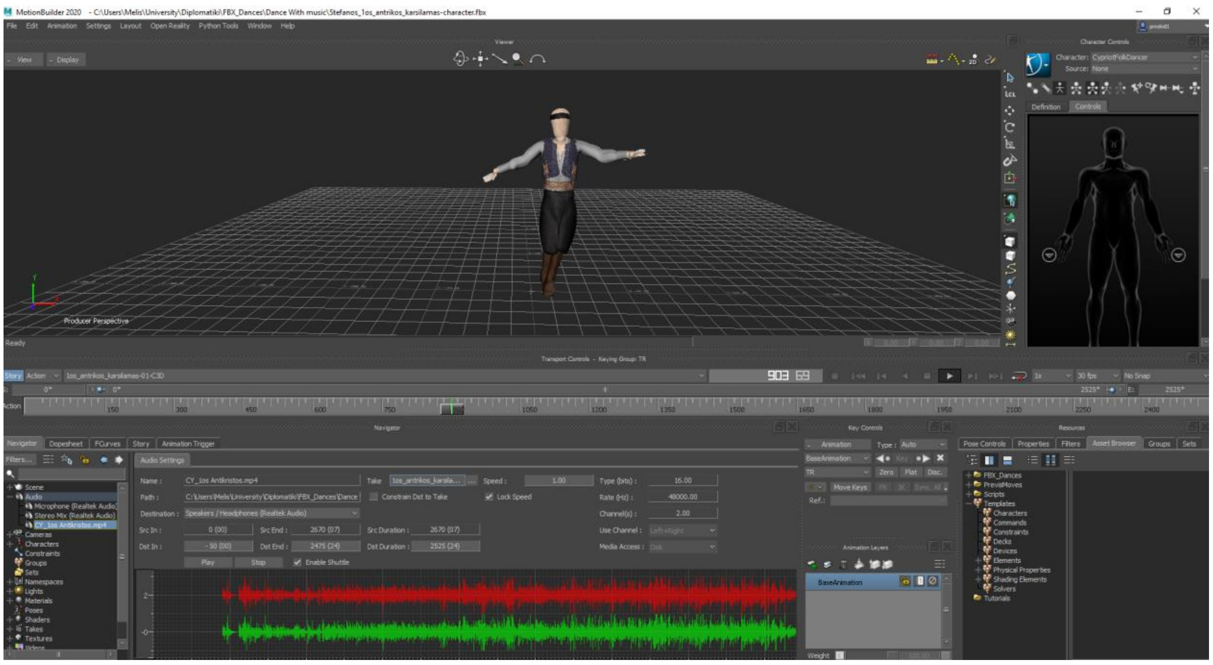


Figure 3.1 Example of using MotionBuilder to synchronize dance with music.

3.2 User’s Movement

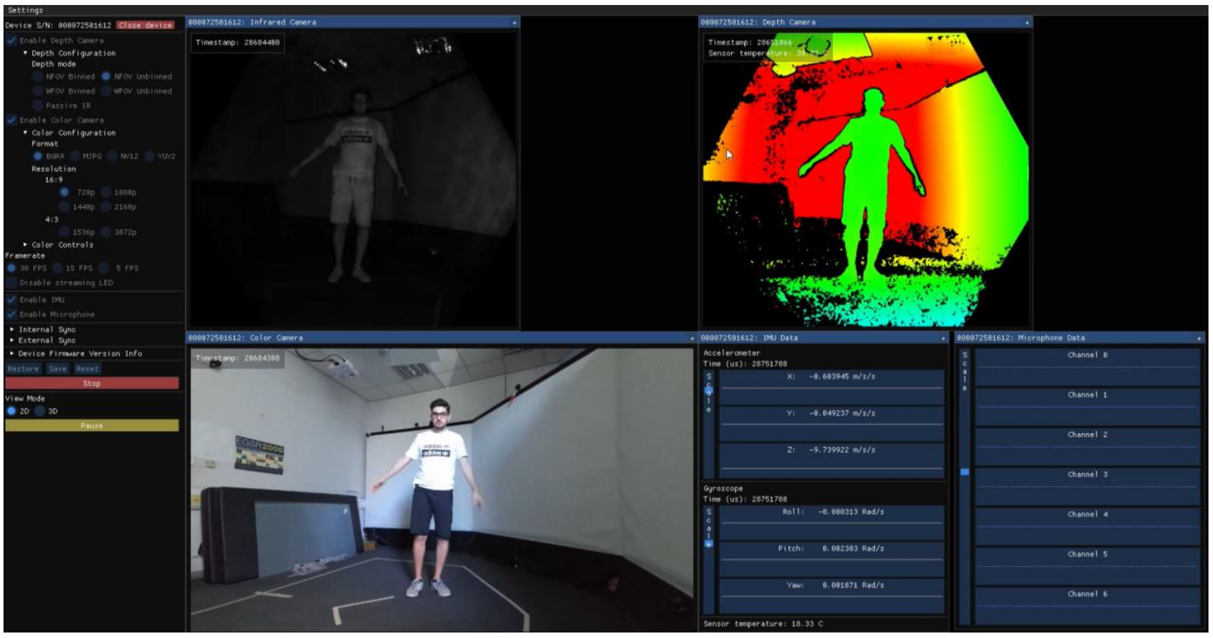
The movement of the user, unlike the teacher’s, needed to be captured in real-time. So, we needed a system that will capture the user’s movement, therefore Microsoft Azure Kinect was used in this platform. More specifically, the user’s performance is built around Kinect’s body tracking SDK feature.

Let us start by looking at what Microsoft Azure Kinect is and how it is used in this platform, and then we will look at how to acquire data from it and how to compare that data with the pre-captured instructor data.

3.2.1 Microsoft Azure Kinect SDK

Azure Kinect DK, Figure 3.2.1.1, is a developer kit with advanced AI sensors that provide sophisticated computer vision and speech models. Kinect contains a depth sensor, spatial microphone array with a video camera, and orientation sensor as an all in-one small device with multiple modes, options, and software development kits (SDKs). The Azure Kinect DK development environment consists of three SDKs: the Sensor SDK (Figure 3.2.1.1), the Body Tracking SDK (Figure 3.2.1.2), and the Speech Cognitive Services SDK which enables microphone access and Azure cloud-based speech services.

Figure 3.2.1.1 Microsoft Azure Kinect, ©Microsoft taken from https://docs.microsoft.com/en-us/azure/kinect-dk/about-azure-kinect-dk



```
Settings
Device S/N: 000072501612  Close device
[✓] Enable Depth Camera
  ▾ Depth Configuration
    Depth mode
      ○ NFOV Binned   ● NFOV Unbinned
      ○ WFOV Binned   ○ WFOV Unbinned
      ○ Passive IR
[✓] Enable Color Camera
  ▾ Color Configuration
    Format
      ● BGRA  ○ MJPG  ○ NV12  ○ YUY2
    Resolution
      16:9
        ● 720p   ○ 1080p
        ○ 1440p  ○ 2160p
      4:3
        ○ 1536p  ○ 3072p
  ▸ Color Controls
Framerate
● 30 FPS  ○ 15 FPS  ○ 5 FPS
[ ] Disable streaming LED
[✓] Enable IMU
[✓] Enable Microphone
▸ Internal Sync
▸ External Sync
▸ Device Firmware Version Info
Restore  Save  Reset
Stop
View Mode
● 2D  ○ 3D
Pause

000072501612: Infrared Camera
Timestamp: 28684400

000072501612: Depth Camera
Timestamp: 28651066
Sensor temperature:

000072501612: Color Camera
Timestamp: 28684388

000072501612: IMU Data
Accelerometer
Time (us): 28751708
X:   -0.603945 m/s/s
Y:   -0.049237 m/s/s
Z:   -9.739922 m/s/s
Gyroscope
Time (us): 28751708
Roll:   -0.000313 Rad/s
Pitch:   0.002303 Rad/s
Yaw:     0.001871 Rad/s
Sensor temperature: 18.33 C

000072501612: Microphone Data
Channel 0
Channel 1
Channel 2
Channel 3
Channel 4
Channel 5
Channel 6
```

Figure 3.2.1.2 Example of the Sensor SDK, in Kinect Viewer.

The Sensor SDK has a Depth camera access and mode control, a RGB camera access and control, a motion sensor access, which includes gyroscope and accelerometer. It also includes a streaming of synchronized Depth-RGB cameras with a configurable delay between them. External system synchronization control with a latency offset between devices that can be configured, and finally a camera frame meta-data access for image resolution and a device calibration data access.

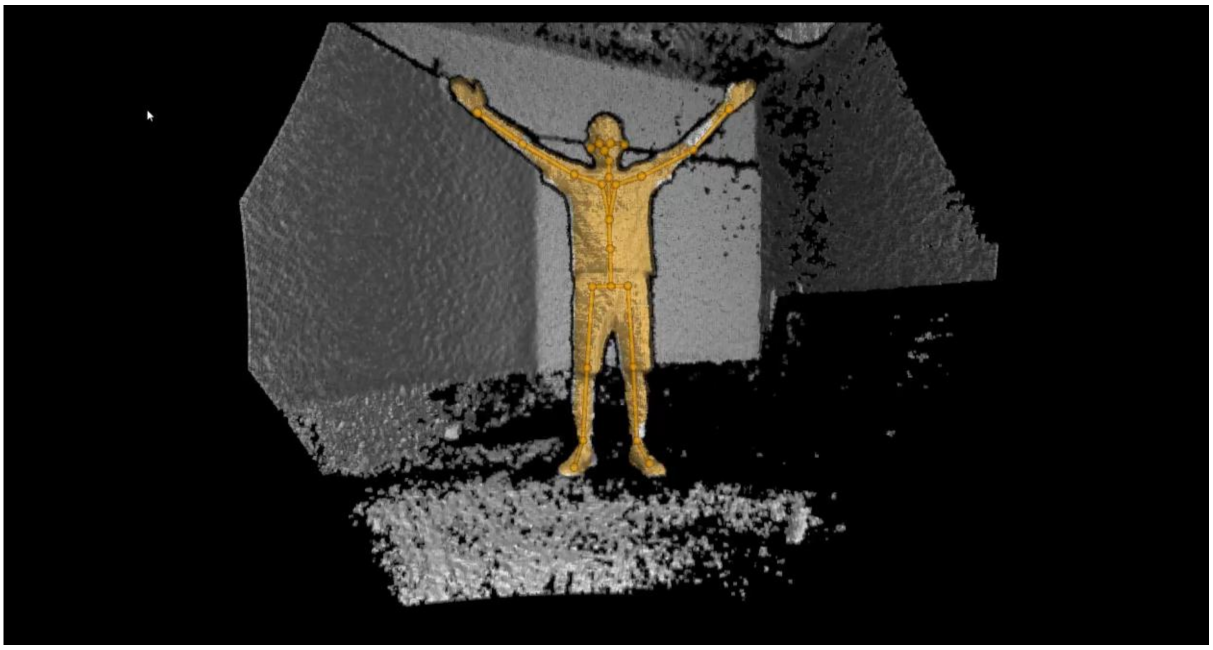


Figure 3.2.1.3 Example of the Body Tracking SDK, in 3D simpler viewer.

The body-tracking provides body segmentation, each partial or even complete body in FOV has an anatomically correct skeleton, it also offers a unique identity for each body and can track bodies over time.

However, the body tracking need specific hardware requirements to work. The recommended minimum Body Tracking SDK configuration for Windows, is Seventh Get Intel® CoreTM i5 Processor (Quad Core 2.4 GHz or faster), a 4 GB Memory, NVIDIA GEFORCE GTX 1050 or equivalent, Dedicated USB3 port. These details were obtained from Microsoft’s Kinect documentation.

Furthermore, the Unity Asset store has a few non-free Kinect Assets, with many functionalities and tools, as for example detecting T-POSE, hand cursor control and other gestures of the user. These assets can make Kinect work, with AMD, NVIDIA and INTEL GPUs. I did not use any of these assets in this platform, due to their cost, but in a future work will be very helpful.

Microsoft provides developers with a software, called Azure Kinect Viewer, which can be used to visualize all device data for Sensor SDK. For example, it can help to verify that sensors are working correctly. The software of the Kinect Viewer can been seen in Figure 3.2.1.2. Finding the perfect position to place the Kinect was not easy, thankfully Azure Kinect Viewer can help finding the perfect position, so the Kinect can capture all the joints of the user. As a result, this software gives you the chance to experiment with camera settings, as well to record and playback your recordings. A similar software, from Microsoft, for Body Tracking SDK is the 3D viewer, which can be seen in action in Figure 3.2.1.3 and Figure 5.1.1.

3.2.2 Azure Kinect body tracking joints

Azure Kinect body tracking can track multiple human bodies at the same time. Each body includes an ID for temporal correlation between frames and the kinematic skeleton. Joint position and orientation are estimates relative to the global depth sensor frame of reference. The position is specified in millimeters. The orientation is expressed as a normalized quaternion. In our case, we only track one body.

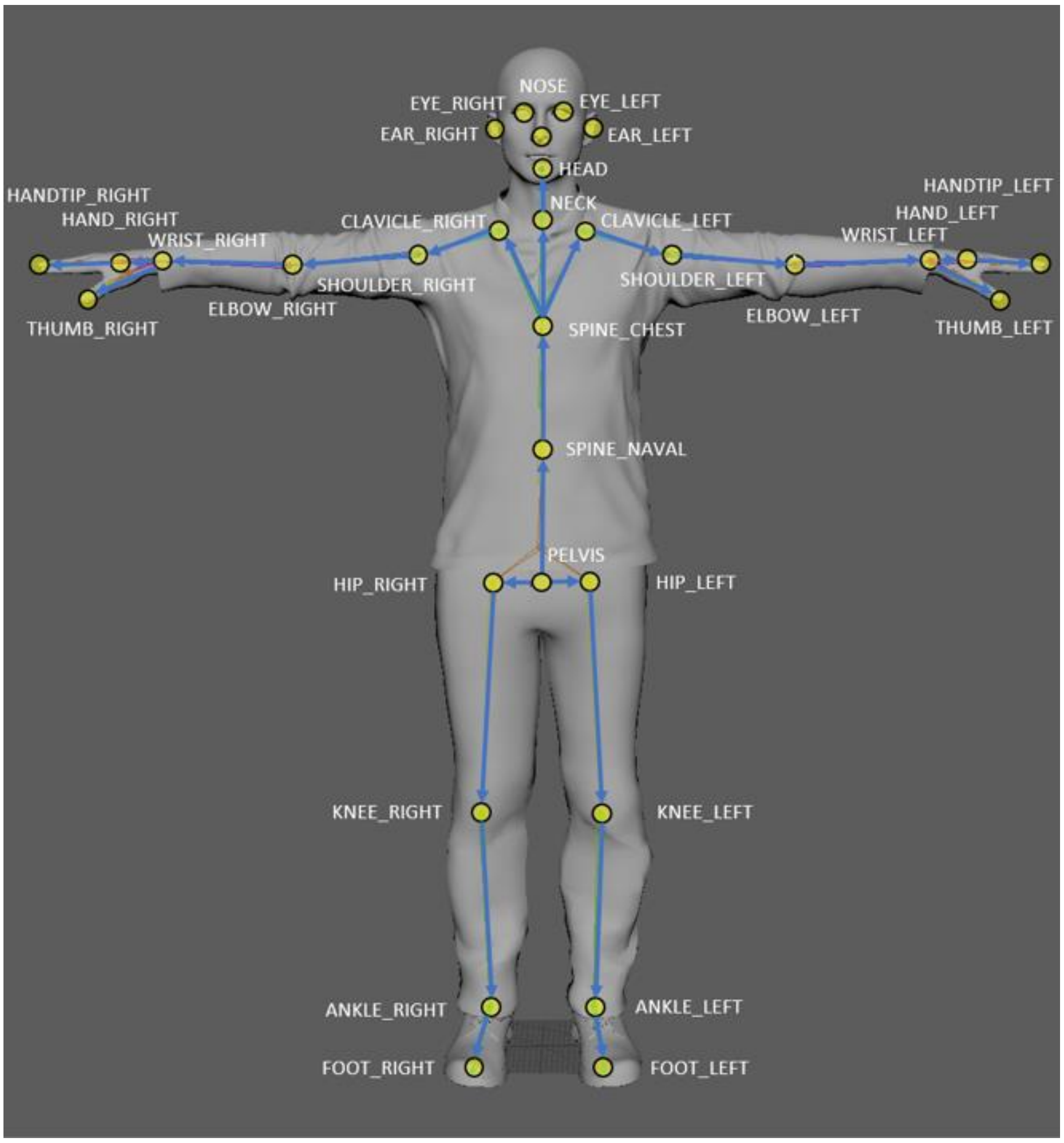


Figure 3.2.2 This figure illustrates the joint locations and connection relative to the human body, ©Microsoft taken from https://docs.microsoft.com/en-us/azure/kinect-dk/body-joints .

3.2.3 Creating the body tracking index map

The body index map includes the instance segmentation map for each body in the depth camera capture. Each pixel maps to the corresponding pixel in the depth or IR image. The value for each pixel represents which body the pixel belongs to. The requirement for a tracker class was applied first to construct this index map in Unity. This body tracking index map will be used to track the transformation of the user. This class also initializes the body tracking system, that renders the skeleton of the user. Creating the skeleton was the first step, the next step was to animate the user’s skeleton to an avatar. The explanation for this is so that the user can see himself dressed as a typical character, such as the tutor, and see a reflection of himself dancing, as seen in Figure 3.2.3.

To enable the Kinect body tracking system, all we had to do is to create a tracker according to Kinect's documentation. Note that, Kinect provides packages for Unity through Visual Studio, those packages are for Sensor and Body Tracking systems which are under the namespaces Microsoft.Azure.Kinect.Sensor and Microsoft.Azure.Kinect.BodyTracking. These packages are working as APIs, Microsoft also gives a documentation for all the classes and functions that are in these APIs. Afterwards, this tracker will store all the joints of the user and these joints will be used to animate a dressed character so, that the user will be able to see himself. More specific, I used a prefab, that contains the component of Tracker script, as seen in Figure 3.2.3.1. When the start function is called, that means when the game object, that holds the specific script, is enabled, at the first frame, the tracker is initialized. The joints that are stored are built in the prefab, so a skeleton is created. Within the update function, which is called in every frame, the skeleton is updated from the joints that the Kinect reads.

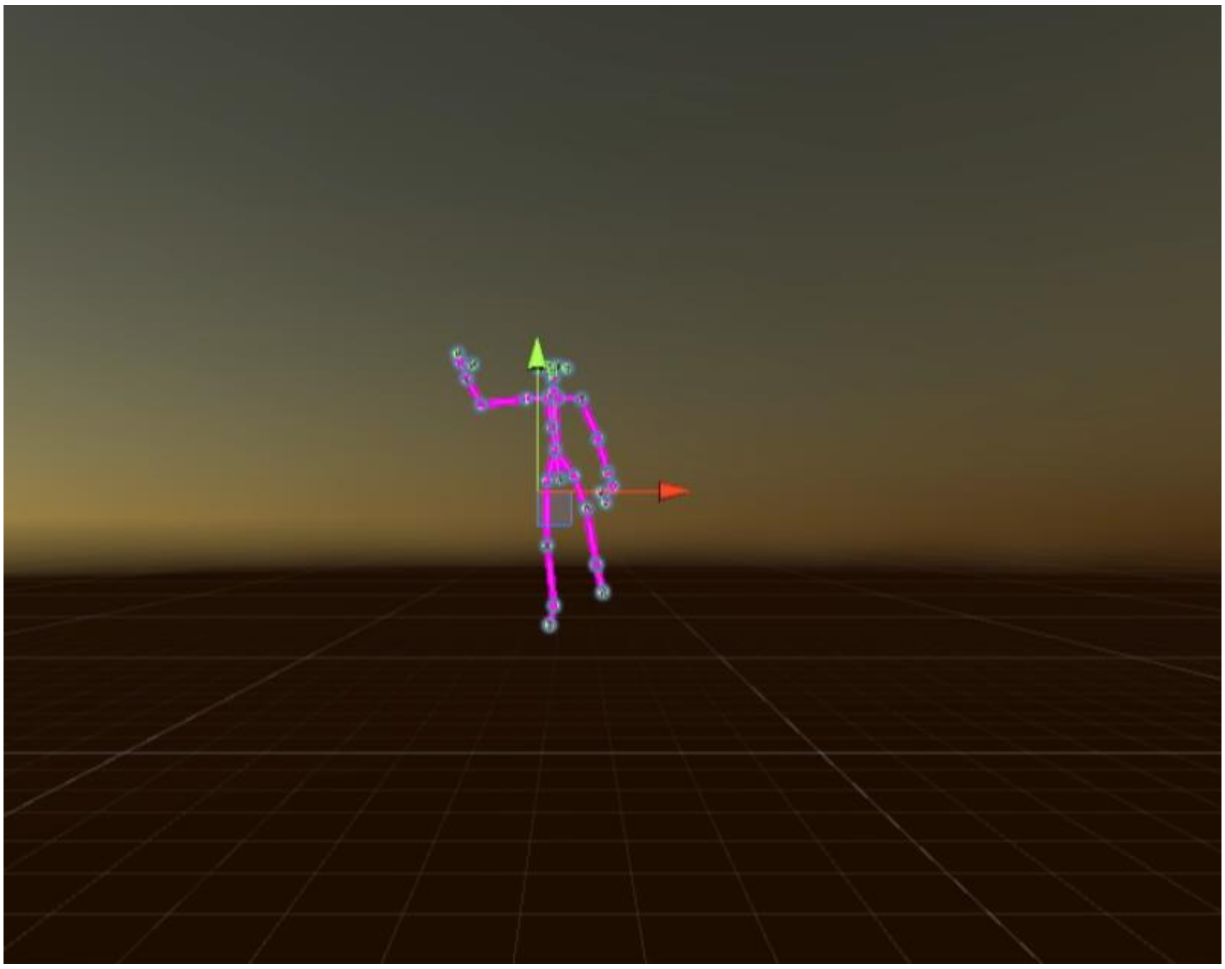


Figure 3.2.3.1 Example of the tracker generated from Kinect.

Algorithm 1 Tracker

```
1. procedure START
2.     tracker ← new tracker()
3. procedure UPDATE
4.     tracker.UpdateTracker()
```

The creation of the puppet, the animated character, is very important as it maps all the Kinect’s joints to an avatar. Having the joints from the tracker, all we must do is to set a new character in the scene and find his root, which will contain as a Transform object, the transformation of the trackers root. Therefore, having the root transformation of the character, with the help of animator, we can locate the rest of the character’s joints, which would have the same transformation values, as the tracker has. The reason we want to create and render the animated user in a canvas, it is so he can see himself while dancing, which is like standing in front of a mirror, just like a real-life dance school. Many dance schools have a large mirror, which is very helpful, so the students can recognize any of their mistakes while dancing. This helps the user experience of the user as well as the user interface looks better.

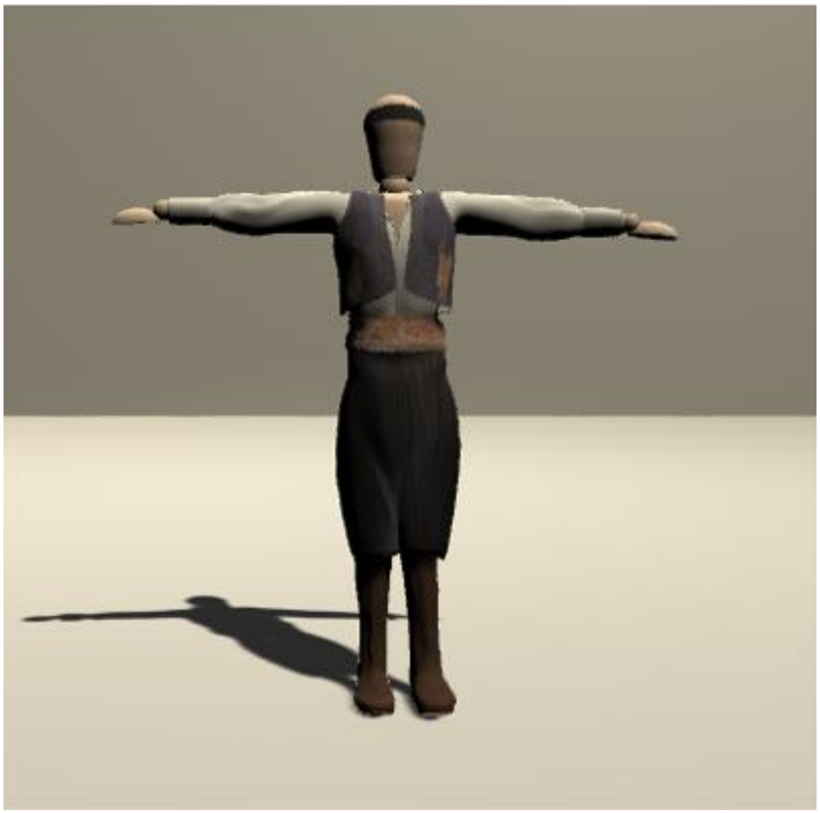


Figure 3.2.3.2 This figure illustrates the characterized avatar, which is the tutor and the student.

Algorithm 2 Puppet Avatar

```
1. procedure START
2. animator ← GetComponent()
3. for i=0 to JointID.Count do
4.     MapKinectJoint(i)
5.     Transform ← Animator
6.     Offset ← GetSkeletonBone
7. end for
```

As you can see from the above algorithm, to create the puppet effect, apart from the joints and the animator of the character, it is important to include the offset of the character. This is optional in the script, but it is used to place for example the character in the middle of the scene, and in Play mode the character will be transferred to the offset coordinates for x and y. Another important calculation is to Map the Kinect joints, with the help of the function MapKinectJoint(int i). This function contains a case statement, which returns a HumanBodyBone[3] for each integer number is given, the default case is the last bone that was used. The joints that *MapKinectJoint* uses are the following: Hips, Spine, Chest, Neck, Head,

[3] The human bone that is queried.

Left upper leg, Left lower leg, Left foot, Left toes, Right upper leg, Right lower leg, Right foot, Right toes, Left shoulder, Left upper arm, Left lower arm, Left hand, Right shoulder, Right upper arm, Right lower arm, Right hand.

Chapter 4

Methodology

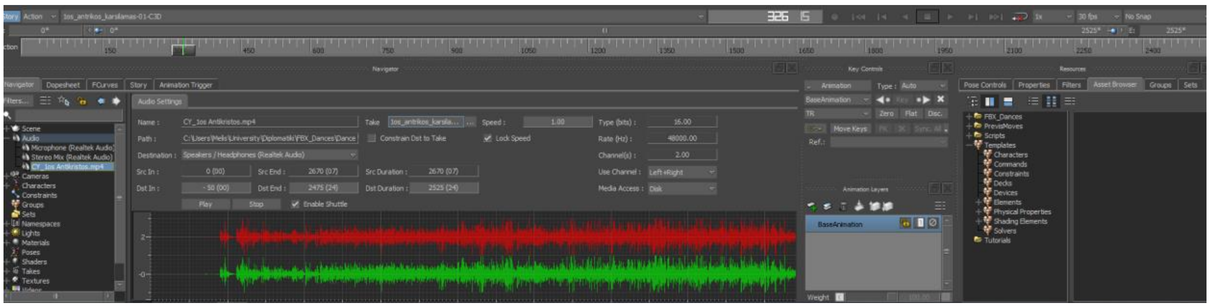
4.1 Music and dance synchronization

Before evaluating the dancer's performance, we must make sure that the dances are synchronized with their music. The reason we want to synchronize the music with the dance is to make it easier for the user to understand the tempo and the rhythm of the dance. Besides, no one can dance without listening to the music, especially if the music is off the tempo.

MotionBuilder was used to accomplish this synchronization, as discussed in Chapter 3.1. It is not easy to get those two components to work together, particularly in MotionBuilder. Many programs exist that can do this, but they are not free. To achieve synchronization in MotionBuilder, I followed these steps: open the motion file that contains the animation clip, then import the audio file, which requires the QuickTime library to be downloaded. Then, after the audio has been imported, you can set the *Dist In* option of the audio in the audio window, which is when the music in the above animation will begin. The audio window in MotionBuilder can be seen by Figure 4.1.1.

Any dance animation begins with a T-POSE, much as the performer did when they were captured. As a result, unlike the music, which is an mp3 file and plays instantly, the performer does not start dancing from the first second. Thankfully, the graphics lab, has saved the footage of the dancers in action, in the dance database. The footage also has background music, which makes it easier to determine when the music should be played. I measure the offset, the delay, between music and the movement for each dance animation and store it as float values, in the code. The reason I did this is to assess the user's success when the dance is in progress, that is while the music is playing, so he can dance with the rhythm.

In more details, we must find the specific frame that the music kicks in, as seen from Figure 4.1.1. The animation is displayed in 30fps[4], so we divide that frame with the number of the fps. The result gives as the offset in seconds. However, MotionBuilder allows you to adjust the number of frames per second (fps), which in our case is 30. For each dance, this offset is calculated, with the above method.



Audio Settings

Name : CY_1os Antikristos.mp4
Take : 1os_antrikos_karsla...
Speed : 1.00
Type (bits) : 16.00
Path : C:\Users\Melis\University\Diplomatiki\FBX_Dances\Dance
Constrain Dst to Take
✓ Lock Speed
Rate (Hz) : 48000.00
Destination : Speakers / Headphones (Realtek Audio)
Channel(s) : 2.00
Src In : 0 (00)
Src End : 2670 (07)
Src Duration : 2670 (07)
Use Channel : Left+Right
Dst In : - 50 (00)
Dst End : 2475 (24)
Dst Duration : 2525 (24)
Media Access : Disk
✓ Enable Shuttle

Figure 4.1.1 Example of synchronizing music with dance.

When the animation begins to play, the music must wait for a specific amount of time, this amount is the offset, the delay, that was calculated before. So, to create the delay functionality, Coroutines were used, as seen from Algorithm 3. A coroutine is like a function that has the ability to pause execution and return control to Unity, but then to continue where it left off on the following frame. By default, a coroutine is resumed on the frame after it yields, but is also possible to induce a time delay using the function *WaitForSeconds*. Coroutines were also used in the 4.2 Dance Evaluation, so that the evaluation will happen in each second.

Algorithm 3 Wait For Animation

```
1.  procedure WaitForAnimation(offset)
2.      Play animation
3.      yield WaitForSeconds(offset)
4.      Play music
```

For the music, I used a package from Hellmade. All the sounds were store as Audio Sources. This package helps the developer to adjust background music, and sound levels, as well as fade in and fade out, as it takes the sound as Sound clips. Each time the music starts, I use a fade in effect that takes approximately one second for the music to reach the highest volume. For the fade out effect I set it to take two seconds, to reach the volume level zero. The same logic is applied in the section where the user must choose a dance, where a preview of the dance and the music background is displayed.

[4] Frames per second

4.2 Dance Evaluation

After rendering Kinect to the Scene, and having the animated teacher synchronized with the music, all it is left is the dance evaluation and comparison. The dance evaluation, in this application includes the local translation of tutor and user, as well as the velocity of the joints. We did not measure the rotation of the user, because the skeleton that is rendered from the Kinect, is a little shaky at the hands, therefore it will be difficult to determine a correct rotation. The reason of this is that Kinect, is a low resolution motion capture system, so there is no point of taking in account the rotation of each joint.

We want to measure the velocity, which was done by using the Algorithm 4, and local location of each bone and finally we want to see if the user's pace and accuracy are similar to the tutor's, indicating that the user is dancing correctly. The local position of a joint means the position of the joint in the scene graph. More specifically, to calculate the local positions for all the joints, I used the vector magnitude function as shown by Algorithm 4. To calculate for example the local position of the Hands, I measured the magnitude of the Hands and the Shoulders, in more details, for the right hand we need the magnitude of the right shoulder and the right hand, for the left hand we need the magnitude of the left shoulder and left hand. A similar logic is applied for the legs, where I use the feet with the hips.

Algorithm 4 Vector Magnitute

```
1.  procedure VectorMagnitute(Vector3 A, Vector3 B)
2.  return     sqrt((A.x – B.x) ^ 2 + (A.y – B.y)^ 2 + (A.z – B.z)^2)
```

The evaluation, and the comparison in general, of the dance takes place in both game modes. Every second, the evaluation is determined. As mentioned in section *4.1 Music and dance synchronization*, coroutines were used to implement the functionality for waiting one second before the evaluation.

The evaluation is a function that returns a float result. The score of the user is the product of this method. Since the evaluation is calculated every second, an array of floats with a length equivalent to the animation’s clip duration is initialized until the dance animation is played. For example, the first antrikos antrikristos karsilamas animation has a length of 78.3 seconds, rounding the float to integer gives the length equal to 78, so there will be 78 evaluations, one for each second. Chapter 4.1's previously calculated music and dance synchronization offset, is subtracted from the duration of each dance, so that the dance and the music are perfectly synchronized. The overall result is the average of the float array’s total summary.

Algorithm 5 Calculate Speed

```
1. procedure CalculateSpeed(Vector3 A, Vector3 B)
2. return     VectorMagnitude(A,B) / Time.deltaTime()
```

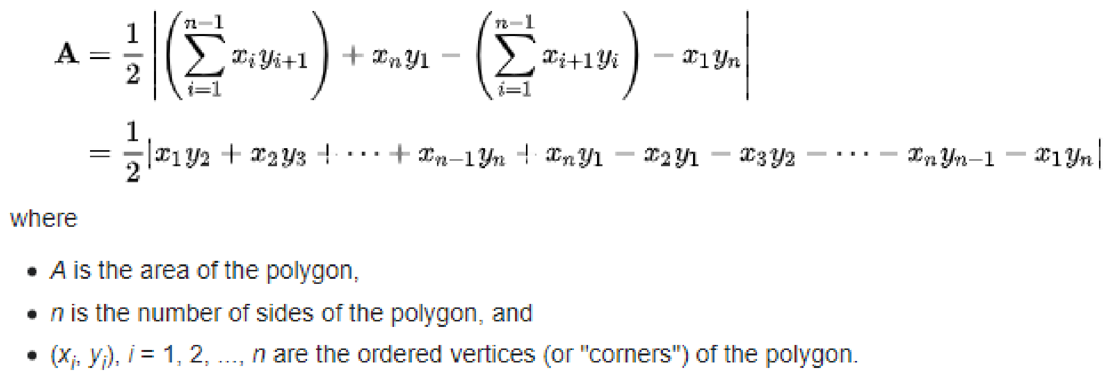
First, we compare the position of the joints of the user and the tutor. There is no need to calculate all the joints in this section; the only ones we need to calculate are the hands (left and right), shoulders (left and right), hips, and feet (left and right). The shoulders and the hips are important so that I can use Algorithm 4 to measure the distance between the hands and shoulders, and the distance between the feet and hips.

Secondly, in the algorithm I also calculate the Shape component. The Shape component, is the way the body is changing towards some point in space, is also used in the calculations. More specifically, shape components features are the following:

- Volume: The bounding volume of all joints, gives the volume of the performer’s skeleton.
- Torso Height: Calculating the crouch, is achieved by measuring the distance between the head and root joints.
- Hands Level: The position of the hands indicates whether they are over the head, between the head and chest, or below the chest.

From the Shape component the Hands Level and Torso Height can be calculated from the first part of the algorithm, which is just a magnitude calculation. The second part of the algorithm is to calculate the Volume.

The Volume is calculated, by calculating the volume of the character. To keep the evaluation simple, for the volume calculation we will need only the five main joints of the character, which are the Head, Hands and Feet. Connecting these joints will create an irregular polygon, so by calculating in each second the volume of this polygon, we know what the volume of the character is. The formula for this equation is based on the Shoelace Theorem, which states that we can measure the area of an irregular polygon using the vertices' coordinates, as seen in Figure 4.2.1. After finding the area from this formula we multiply with the length on the Z axis. Finding the volume of a polygon requires to multiply the base with the height. In our case the base is the area that the user creates, and the height, is the distance of the minimum and maximum value on the Z axis, the depth of the user.



$$\mathbf{A} = \frac{1}{2}\left|\left(\sum_{i=1}^{n-1} x_i y_{i+1}\right) + x_n y_1 - \left(\sum_{i=1}^{n-1} x_{i+1} y_i\right) - x_1 y_n\right|$$

$$= \frac{1}{2}|x_1y_2 + x_2y_3 + \cdots + x_{n-1}y_n + x_ny_1 - x_2y_1 - x_3y_2 - \cdots - x_ny_{n-1} - x_1y_n|$$

where

- *A* is the area of the polygon,
- *n* is the number of sides of the polygon, and
- (x_i, y_i), $i = 1, 2, \ldots, n$ are the ordered vertices (or "corners") of the polygon.

Figure 4.2.1 This is the Shoelace Theorem, that calculates the area of an irregular polygon, taken from https://en.wikipedia.org/wiki/Shoelace_formula

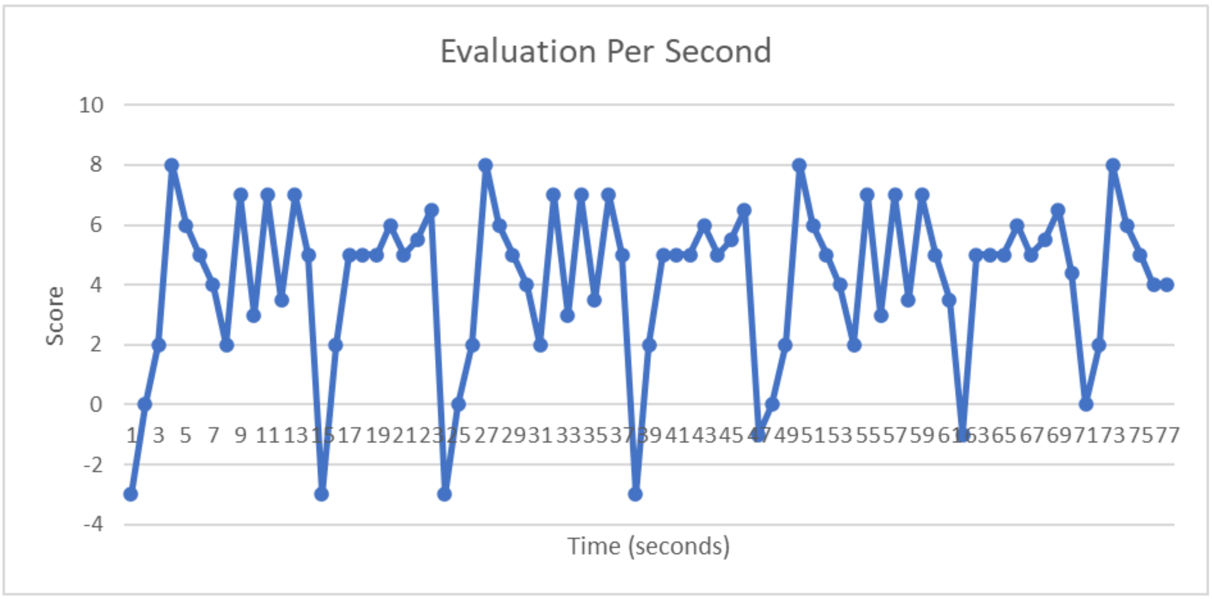
Of course, the result of the volume will once again be the result of the comparison of both user's and teacher's volume. The fraction of the result, similarly as what we did before, will determine how close the two volumes are. The same principles will be used here. If the result is between 60% and above, then the score of the user will increase by one, but if the result it is lower that 60%, then the result will decrease by one.

Algorithm 6 Evaluation

```
1.  procedure float Evaluation()
2.  score ← 0
3.  Get last positions of hands, shoulder, hip and feet of student and tutor.
4.  Find the position of hands, shoulders, hip and feet of student and tutor.
5.  Calculate the speed of each joint, using last and current position using Algorithm 5.
6.  score ← Compare speeds for each joint of user and tutor.
7.  Find the distances for user and tutor using Algorithm 3.
8.      VectorMagnitude(left_feet, hip)
9.      VectorMagnitude(right_feet, hip)
10.     VectorMagnitude(left_hand, left_shoulder)
11.     VectorMagnitude(right_hand, right_shoulder)
12. score ← Compare the distances of user and tutor.
13. Calculate the volume of user and tutor.
14. score ← Compare the volume of user and tutor.
15. save_positions()
16. return score;
```

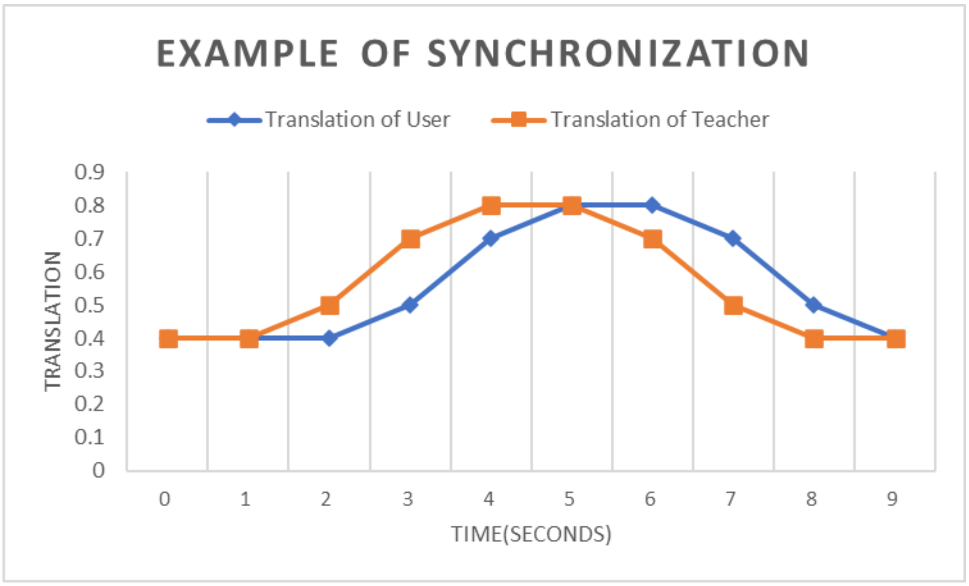
As we can see from the Algorithm 6, at lines 6 and 12 we have to compare user's and tutor's data. Here, the main idea is to divide the user's current data with the tutor's and get the result of the fraction. The fraction will show, how close the two numbers are. The score increases by one, if the result of the fraction is between 0.60 and 1.0. That means that if the user in the current second of the evaluation, is close to teacher's movements by 60% and above then he increases his score by one point. If the result of the fraction is lower than 60% the score decreases by one. That is, if a user fails in a second, the score for that second is likely to be negative. In this algorithm, all comparisons, including speed, distance, and volume, have the same weight in the final result since they add and deduct one point. This method is a balanced assessment, and it is based on the same criteria as all of the contrasts.

An example of how this evaluation works, specifically the score, is shown in Graph 4.2.1, for the 1os antrikos antrikristos karsilamas dance. For each second, we have a value of the total score, as said these values are stored in an array, where in the end of the dance the average score of these scores is calculated, giving the result.



Graph 4.2.1 Example of the Evaluation per second.

The evaluation could be calculated different, by making sure that the user and the teacher are fully synchronized. This is much more difficult to calculate and implement, because for each component, in a high-level description, you must find the peaks of each value and compare them for both user and teacher. For example, if the teacher is standing still for 2 seconds, and moves the right hand, then the transformation of the right hand will be still for 2 seconds, and on the third second will increase. If the user tries to imitate this move will probably do that on the fourth second. Comparing those two peaks, is a way to compare the same movements of user and teacher, no matter if the user is not fully synchronized with the teacher, as shown in Graph 4.2.1.



Graph 4.2.2 Example of the synchronization that could be used.

As previously said, this synchronization is much more complicated than the one I used. In fact, achieving this synchronization would take a long time, for understanding how it works and implementing it.

As a result, the logic that I used is much simpler than this, so the user must act at the same time as the instructor in order for his or hers results to be properly calculated. This, however, makes the gameplay more intimidating for amateur users, but it also increases their results as they improve in time.

Note that, this evaluation does not calculate the coordinates of the joints in the scene. To be more specific, it ignores the fact that the teacher’s feet are at position (0.5, 0, -0.25), implying that the user’s feet must be near that location.

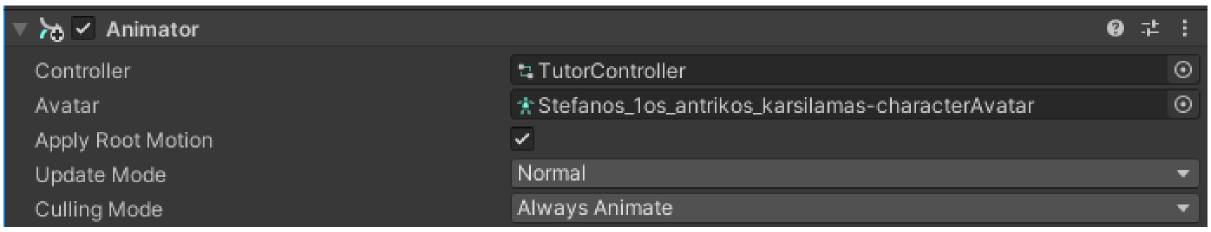
The last part of the evaluation is to display a realistic as possible feedback to the user like a real instructor. This feedback could be, for example a message such as “Perfect dance!”, “Nice try!” or “Bad try!”. Feedback messages are very important in applications, especially in games, because they help user understand, what is going on, and in general are a part of the user experience. Showing messages in the screen is the best way to give user a proper feedback of his or her performance.

4.3 Modes of Gameplay

This platform supports two game modes that make it easier for the user to learn a dance. The first mode is referred to as "Play Dance," while the second is referred to as "Learn Dance". In both modes, there is always the tutor that performs the dance.

The tutor is an fbx motion file, where the animation type must be set to *Humanoid*, and the animator component must have set to true the *apply root* motion, as seen from Figure 4.3.1 and Figure 4.3.2, in order to play the animation clip through the animator component of the character. If neither of these two settings is used, the character will most likely lose his translation and will only be able to move the hands and the legs. As a result, this is not going to look good for the animation.

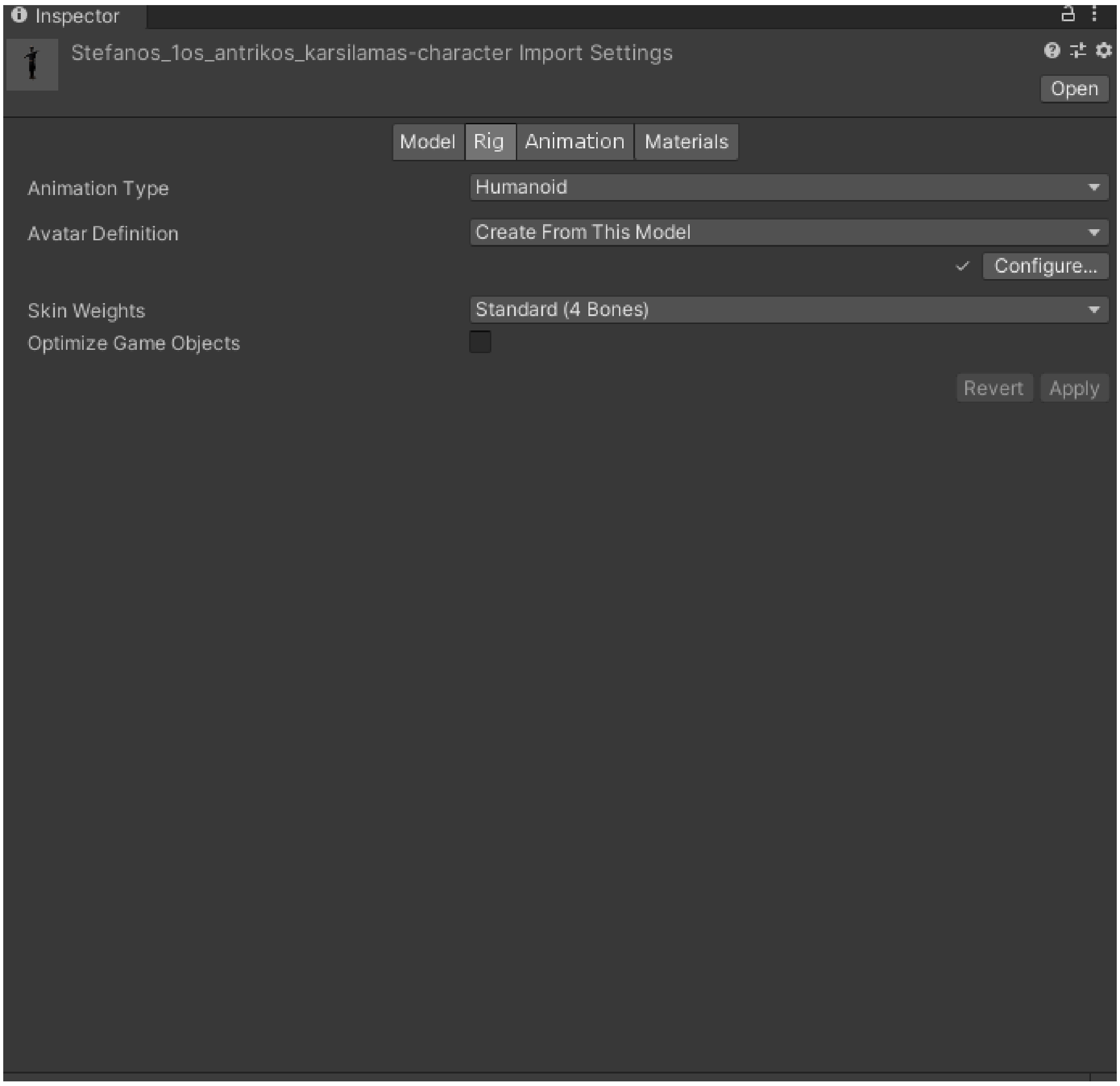
Because the instructor is rotated 180 degrees, and faces the camera, his movements are mirrored. For example, if he raises his left hand, the user will might think that the tutor raised his left hand. To correct this misunderstanding, we added two NPC[5] student dancers, who face the tutor in the same way as the user does. The NPC students dance simultaneously with the tutor. The NPC characters are shorter than the teacher, allowing the teacher to stand out. This method would make it much easier for the user to comprehend the instructor's movements. So, it does not matter if the user watches the tutor or the NPC, while dancing. The importance of the NPC, in this game, is shown in more details in the next chapter, Chapter 5.


Animator

Controller	TutorController
Avatar	Stefanos_1os_antrikos_karsilamas-characterAvatar
Apply Root Motion	✓
Update Mode	Normal
Culling Mode	Always Animate

Figure 4.3.1 Animator component of the character.

[5] NPC: Non-player character


Inspector

Stefanos_1os_antrikos_karsilamas-character Import Settings

Open

Model | Rig | Animation | Materials

Animation Type: Humanoid

Avatar Definition: Create From This Model

✓ Configure...

Skin Weights: Standard (4 Bones)

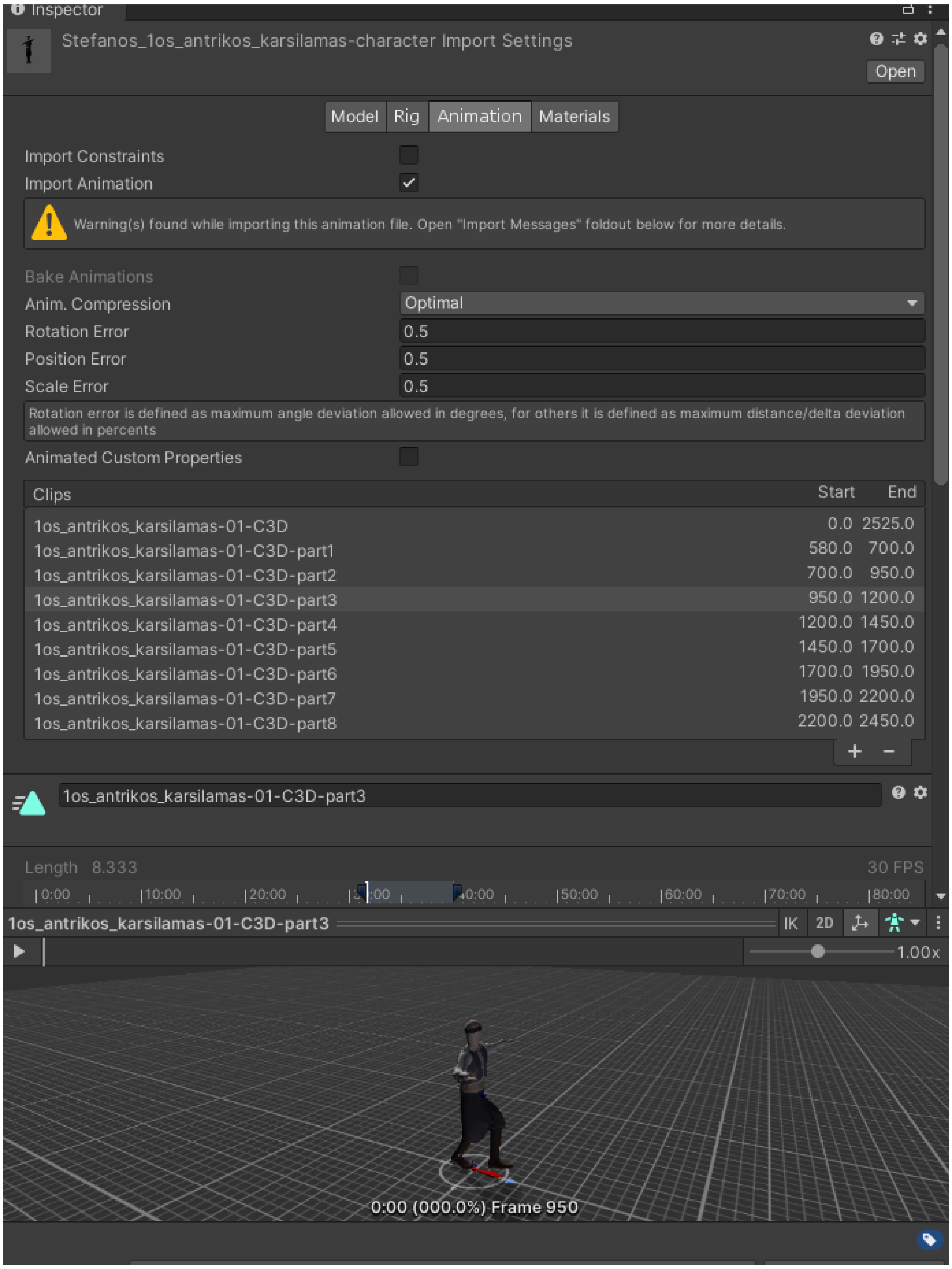
Optimize Game Objects: ☐

Revert | Apply

Figure 4.3.2 Animation Type of the character

In the first mode, the tutor performs the whole dance. It is very difficult, for an amateur to follow the tutor in this mode, as sometimes the dance might be very swift, or might have difficult movements and turns. In this mode, we do not change the animator's speed, in contrast with the "Learn mode". as the part will play only by using the function Animator.Play(), like the Algorithm 3. This mode is recommended for users that they considered themselves as intermediate or experts.

The "Learn Dance" is an excellent choice to start for inexperienced users. The dance is performed in parts. The parts are created manually in the Animation tab of the motion file, see Figure 4.3.3. Each part it is a copy of the original Animation clip, with a different Start and End frame. These Animation clips that are created, are used in the Animator of the activated character, as seen in Figure 4.3.4. Having all those parts in the Animator, it is easier to decide which part will be played. A script TagHolder.cs has all the string of all the animations in the game. When for example the user selects the Zeimbekiko dance, in learning mode, a list gets filled with all the parts of that animation. So, each time a part is successfully performed, the iteration in the list moves to the next one, until it reaches the last part. If the user achieves a score of 50% or higher, that is a successful performance, therefore the tutor can proceed to the next part. This is repeated before the dance reaches its conclusion. After the dance finishes, then the overall evaluation is displayed on the screen. The overall evaluation is the average evaluation of each part.



Inspector

Stefanos_1os_antrikos_karsilamas-character Import Settings

Open

Model | Rig | Animation | Materials

Import Constraints	☐
Import Animation	✓

Warning(s) found while importing this animation file. Open "Import Messages" foldout below for more details.

Bake Animations	☐
Anim. Compression	Optimal
Rotation Error	0.5
Position Error	0.5
Scale Error	0.5

Rotation error is defined as maximum angle deviation allowed in degrees, for others it is defined as maximum distance/delta deviation allowed in percents

Animated Custom Properties	☐

Clips	Start	End
1os_antrikos_karsilamas-01-C3D	0.0	2525.0
1os_antrikos_karsilamas-01-C3D-part1	580.0	700.0
1os_antrikos_karsilamas-01-C3D-part2	700.0	950.0
1os_antrikos_karsilamas-01-C3D-part3	950.0	1200.0
1os_antrikos_karsilamas-01-C3D-part4	1200.0	1450.0
1os_antrikos_karsilamas-01-C3D-part5	1450.0	1700.0
1os_antrikos_karsilamas-01-C3D-part6	1700.0	1950.0
1os_antrikos_karsilamas-01-C3D-part7	1950.0	2200.0
1os_antrikos_karsilamas-01-C3D-part8	2200.0	2450.0

1os_antrikos_karsilamas-01-C3D-part3

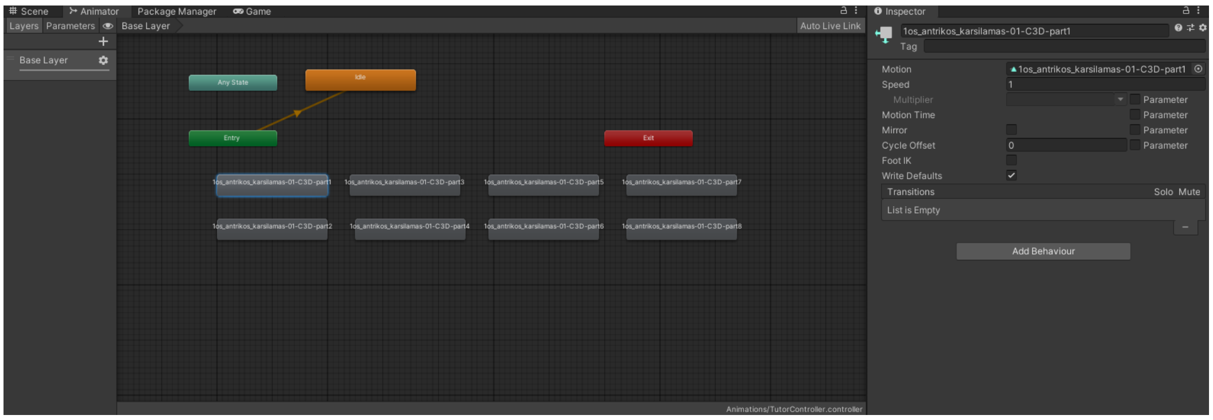
Length 8.333 30 FPS

1os_antrikos_karsilamas-01-C3D-part3

1.00x

0:00 (000.0%) Frame 950

Figure 4.3.3 Example of the Animation tab and the parts of a character in Unity.


Scene
Animator
Package Manager
Game
Layers
Parameters
Base Layer
Auto Live Link
Base Layer
Any State
Idle
Entry
Exit
1os_antrikos_karsilamas-01-C3D-part1
1os_antrikos_karsilamas-01-C3D-part3
1os_antrikos_karsilamas-01-C3D-part5
1os_antrikos_karsilamas-01-C3D-part7
1os_antrikos_karsilamas-01-C3D-part2
1os_antrikos_karsilamas-01-C3D-part4
1os_antrikos_karsilamas-01-C3D-part6
1os_antrikos_karsilamas-01-C3D-part8
Animations/TutorController.controller
Inspector
Add Behaviour

1os_antrikos_karsilamas-01-C3D-part1
Tag

Motion	1os_antrikos_karsilamas-01-C3D-part1	
Speed	1	
Multiplier		Parameter
Motion Time		Parameter
Mirror		Parameter
Cycle Offset	0	Parameter
Foot IK		
Write Defaults	✓	

Transitions Solo Mute
List is Empty

Figure 4.3.4 How the Animation clips are used in the Animator, in Unity.

There is no transition between the animation clips in the Animator window. All we do in this part is to first set animator’s speed to one, then play the animation using the Animator.Play() function and pass to the function the name of the state as a parameter. The name of the state, as mentioned above, is the current name in the list. After the current animation part is finished, the animator’s speed is set to zero. This has as a result to stop the animations that the character is playing, to show a message that the user is moving on to next part. This is a loop that keeps playing until the Animator reaches the last state, which is the last Animation clip.

Chapter 5

Results and Applications

5.1 Kinect Results

Getting data from Kinect was not easy, especially when sometimes the data were wrong. The reason that the data could be wrong were the following errors: The position of the Kinect in the room could not capture the feet, because the user was very close to it. The Kinect was too high and could not read the feet of the user correctly. The Kinect was lower from the user, therefore the Kinect could not read his head. Sometimes the user, could get out of the Field of View of the Kinect while dancing and for those seconds he was missing, Kinect could not read the missing joints, as a result the animated character would "break", as seen from Figure 5.1.1, because the tracker cannot map its joints to the animated character. After many trials and errors, I found a spot in the room where I could place the Kinect and I was sure that it is high was perfect and it could record all of the user's joints. The preview of the user that is rendered to the game play, was very helpful to understand if Kinect misses one of the user's joints, as well as the Kinect Viewer. Despite these errors, the Kinect creates a character, but it is not very smooth. Sometimes, the hands are showing like they are shaking.

As mentioned in the beginning of this work, Kinect can track up to 5 people at the same time. In this platform, Kinect tracks only one person-body. So, the Kinect can lose the data of the user, in case another user appears in the field of view of the Kinect. By losing the data, means that Kinect will return for example for the right had joint the position of (0,0,0), instead of (2,1,0.5), there the evaluation will produce a non-valid result.

Another problem that might affect the Kinect result is the following, if a part of the body gets in front of another, this makes it hard for the Kinect to capture the part that is behind the other. Of course, if this movement happens very fast there is no problem. Something like this, can be seen from Figure 5.1.2, where we can see the user that is crouching, and Kinect cannot capture his Left foot joints.

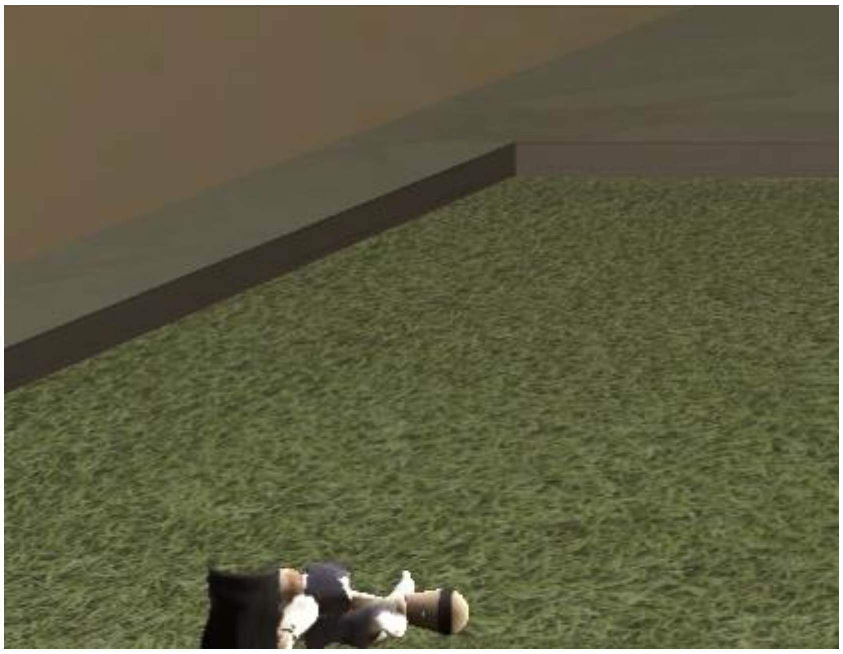


Figure 5.1.1 Example of character moving away from the FOV of Kinect.

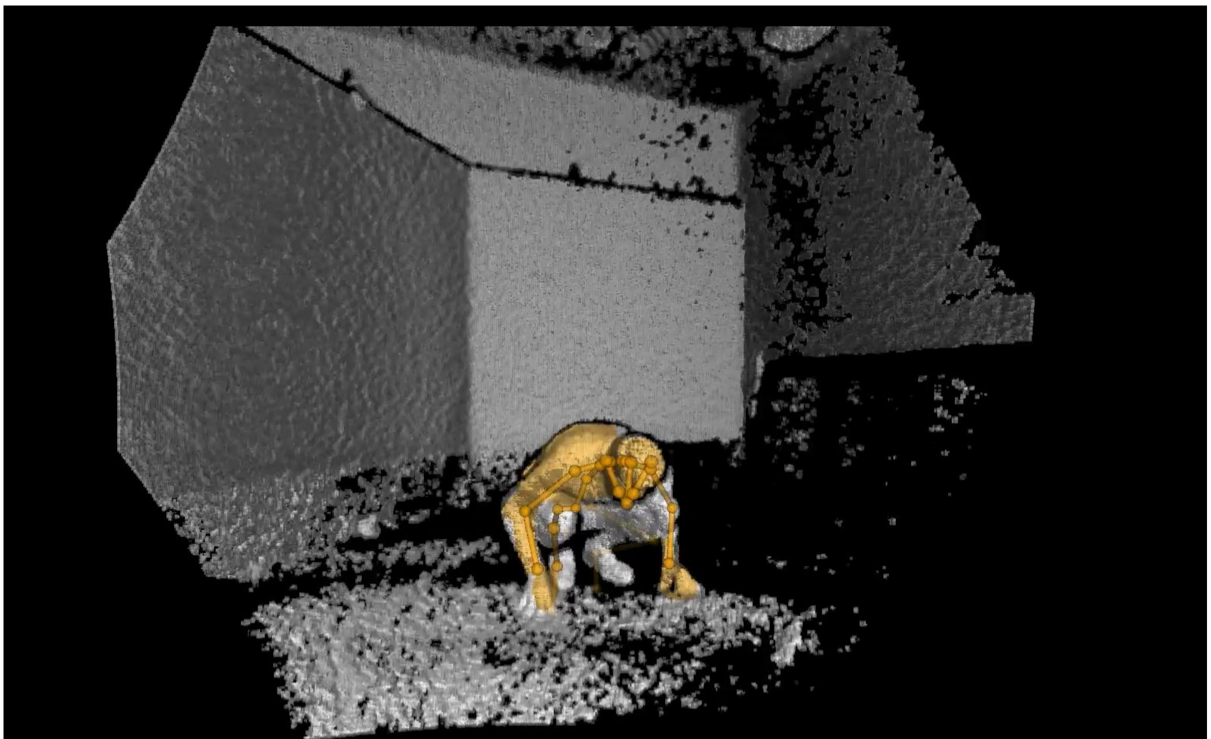
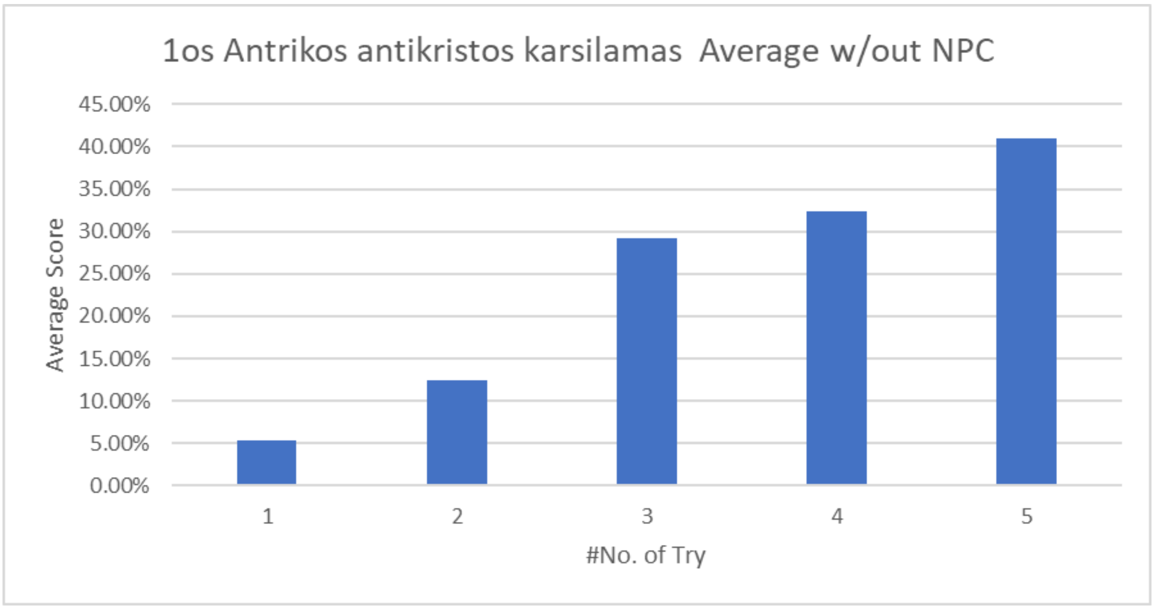


Figure 5.1.2 Example of how Kinect captures the user’s crouching.

After many attempts in different dances, including both modes, it was very difficult for me to keep up with the teacher. The reason of that is the synchronization problem of the user and the teacher. To achieve a perfect evaluation, the user must dance simultaneously with the teacher, therefore this is very difficult especially for someone inexperienced like me. As a result, setting the comparison to 60% was the optimal value. The comparison is defined in *Chapter 4.2 Dance Evaluation*, where the algorithm compares the magnitude, pace, and volume of both the user and the instructor. In addition to considering the user's movement success, these values must be 60% or higher. Another reason that the 60% and above was decided, is that Kinect, as it a low resolution motion capture system, does not calculate the exact position of a joint. This threshold was also reduced from the amount of percentage was required to pass the successful comparison.

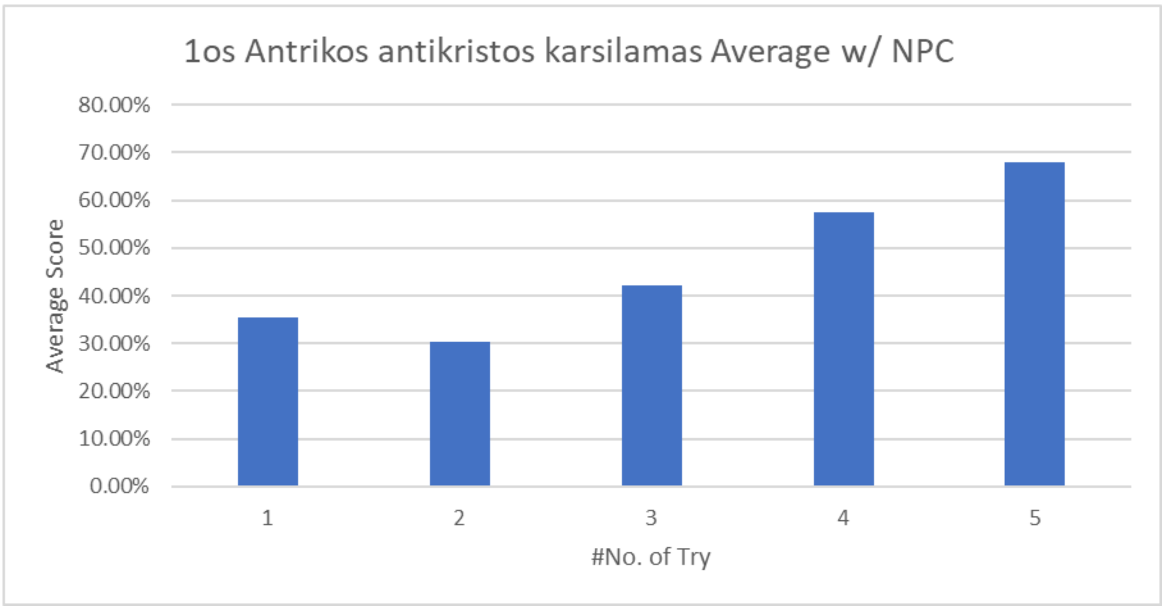
5.2 Results of Play Dance

The Play Dance mode it is not easy, especially for an amateur. The type of dance can be very fast or very tricky, which makes it hard for the user to keep up with the teacher. The only dance, which is easier from the others is the Hasaposerviko. The reason that this dance is easier, than the other, is that it has very simple and easy movements, therefore there are not many turns and kicks. I attempted the first dance, the "1os antrikos antrikristos karsilamas", which for me is an intermediate dance, five times to see how the evaluation works and, in general, if the whole idea will help me memorize sections and develop my skills for the particular dance, as seen in Graph 5.2.1. This experiment was done without the NPC dancers. The average score in this section was just over 40%.



Graph 5.2.1 Results after 5 tries of 1os Antrikos antrikristos karsilamas without NPC dancers.

The NPCs was very crucial as they helped a lot. I tried the same dance, but this time with the NPCs, as seen in Graph 5.2.2. The result is that after five tries, I managed to reach almost 70% of total score. To be honest, I was not looking at the instructor in the dance; instead, I was looking at one of the NPC.

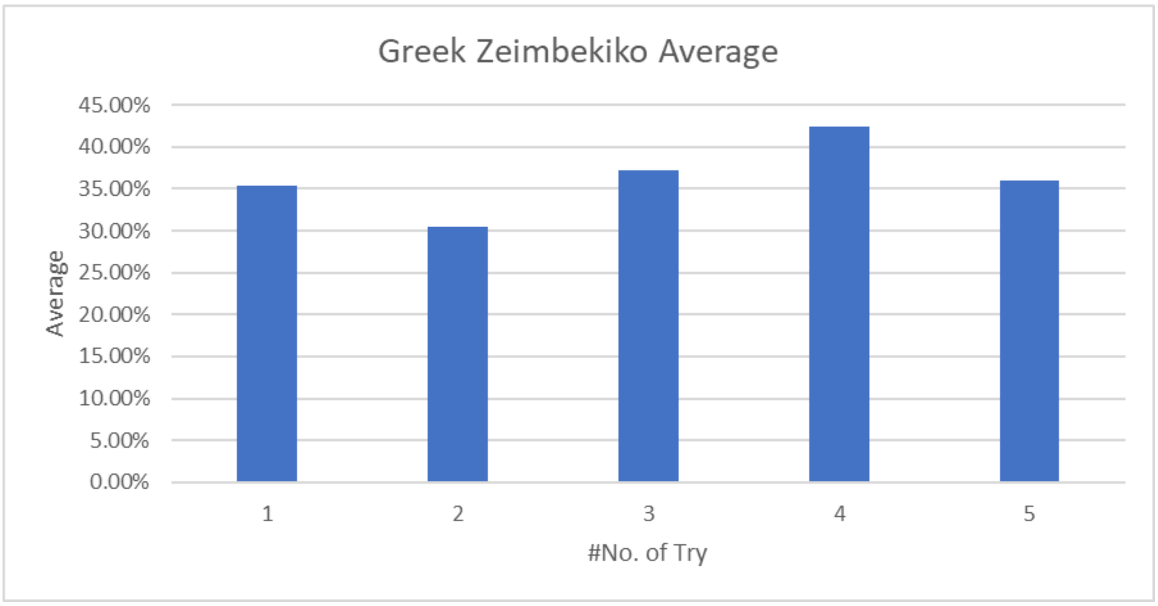


Graph 5.2.2 Results after 5 tries of 1os Antrikos antikristos karsilamas with NPC dancers.

By observing the increase in the average score in the above performance, it is clear that my dance skills have improved. This is a positive outcome because it indicates that we met the aim of our submission.

In addition to the NPCs, the illumination of the teacher and the NPCs was critical. With the perfect illumination the user can see all the details of the teacher.

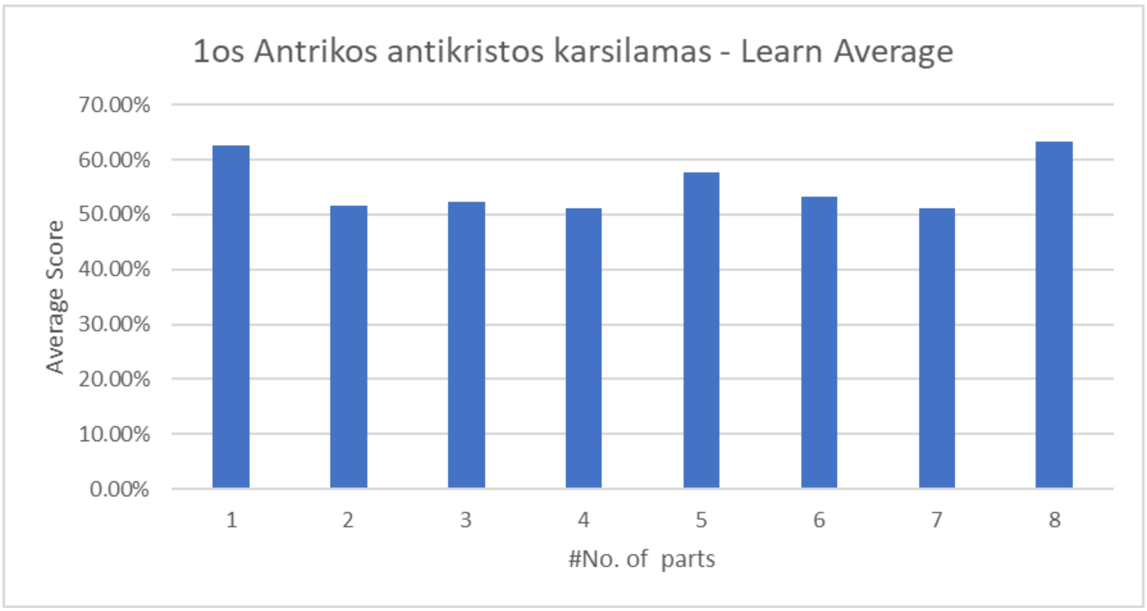
There are a couple of twists in some of the dances, which may be a challenge. The Greek Zeimbekiko, for example, has a section where the instructor dances while turning 180 degrees. That means I, the user, must also turn 180 degrees in order to continue dancing, but I am unable to continue the dance, because I am not facing the computer screen. As a result, my body was facing the other way and I was trying to see the screen, I could not synchronize with the teacher’s movement. This dance has a lot of parts were this happens, that is why the results as seen in Graph 5.2.3, are very disappointing. Maybe, if the dance did had fewer turns, it would be simpler for everyone to dance.



Graph 5.2.3 Results after 5 tries of Greek Zeimbekiko with NPC dancers.

5.3 Results of Learn Dance

The Learn Dance mode is very important, as it is one of our objectives, to split the dance in animation windows. All of the dances in this platform were cut into short animation windows, each lasting between 4-8 seconds. Because the dance is shown in parts, it is easier for a user to keep track with the teacher. As seen in Graph 5.3.1, the results are very promising in this mode, as the way the dance is portrayed aids the user in correctly perceiving the motions. The only issue with this section is dividing the dance into sub-pieces, that is, dividing the dance into sections that cover at least two entire movements, such as turning and kicking, and not stopping the part in the middle of a step.



Graph 5.3.1 Results for 1os Antrikos antrikristos karsilamas in Learn mode.

The final result is the average result of the succeeds parts. For example, if a user fails a part, it will not be calculated to the final result, only the successful parts will be included. The Learn Dance mode, forces user to repeat a part if he or she did not mage to receive a score of 50% and above. This way the user improves his results, as shown from Graph 5.3.2. In this graph, it was the third attempt of me dancing the 1os antrikos antrikristos karsilamas, that is why the result are better from Graph 5.3.1. Again in this mode we ensure that we achieved our objectives, as the result in each try get better and better.

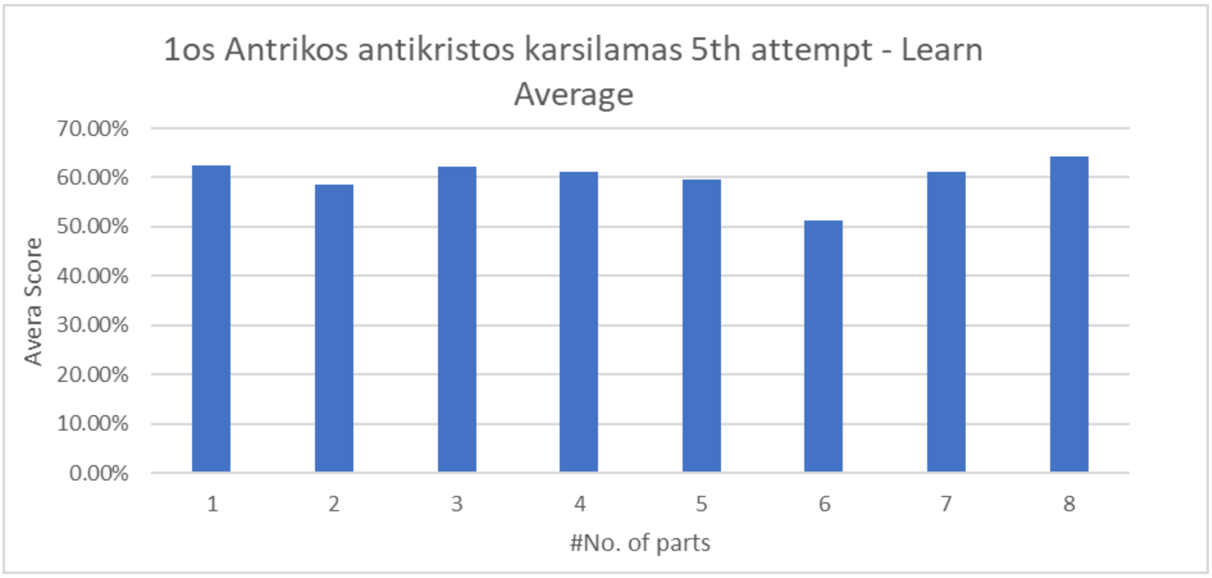


Figure 5.3.1 1os Antrikos antikristos karsilamas in Learn Mode.

Chapter 6

User Interface

6.1 Scenes

The game is made in two scenes. The first scene is the main menu scene, and the second scene is the gameplay scene.

The Menu scene, it is just a canvas, which only has two options “Play” and “Quit”. The first option starts the game and the second one quits the application, as seen in the figure 6.1.1. When the user clicks on Play button, then the gameplay scene is activated, and the game begins.

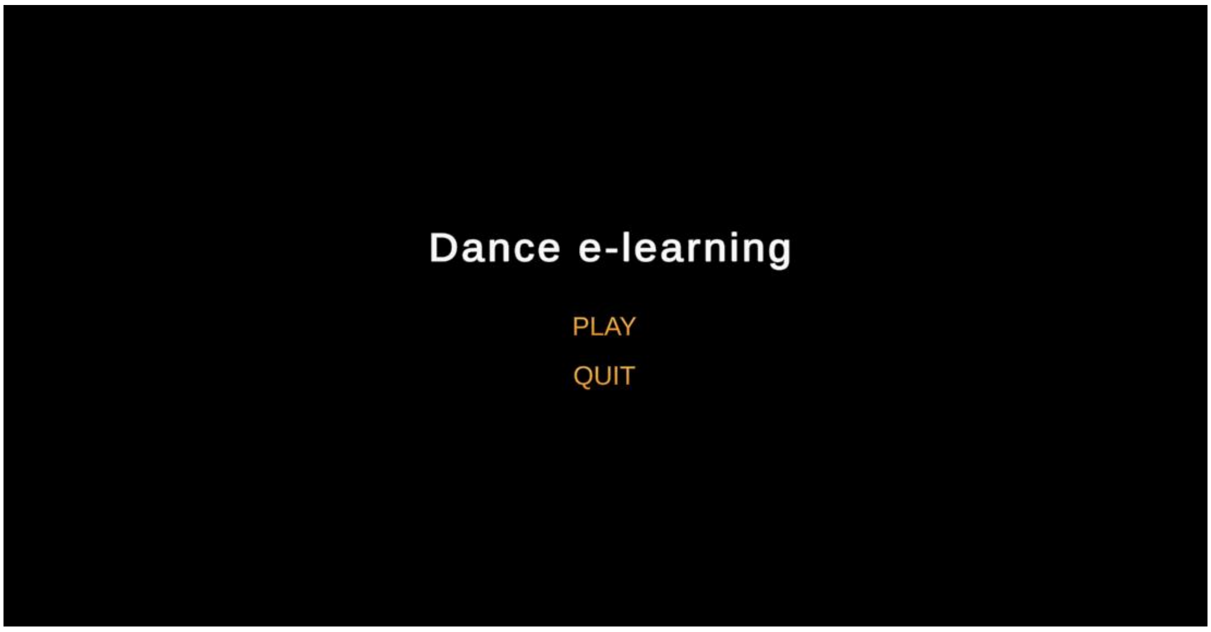


Figure 6.1.1 Main menu scene.

There is just an old mansion and the characters in the gameplay scene as shown in Figure 6.1.2. The characters, in this scene are the intro dancer, the avatar that is used for previewing the dances, and the instructor with two students, as well as the user who is tracked from Kinect. The user is navigated through the house with an animated camera package called Cinemachine. This type of navigation was chosen to make it easy for the player to use the game's functionality, as well as to allow the user to look around the house, which makes the game more realistic, and gives a cinematic touch to the game.

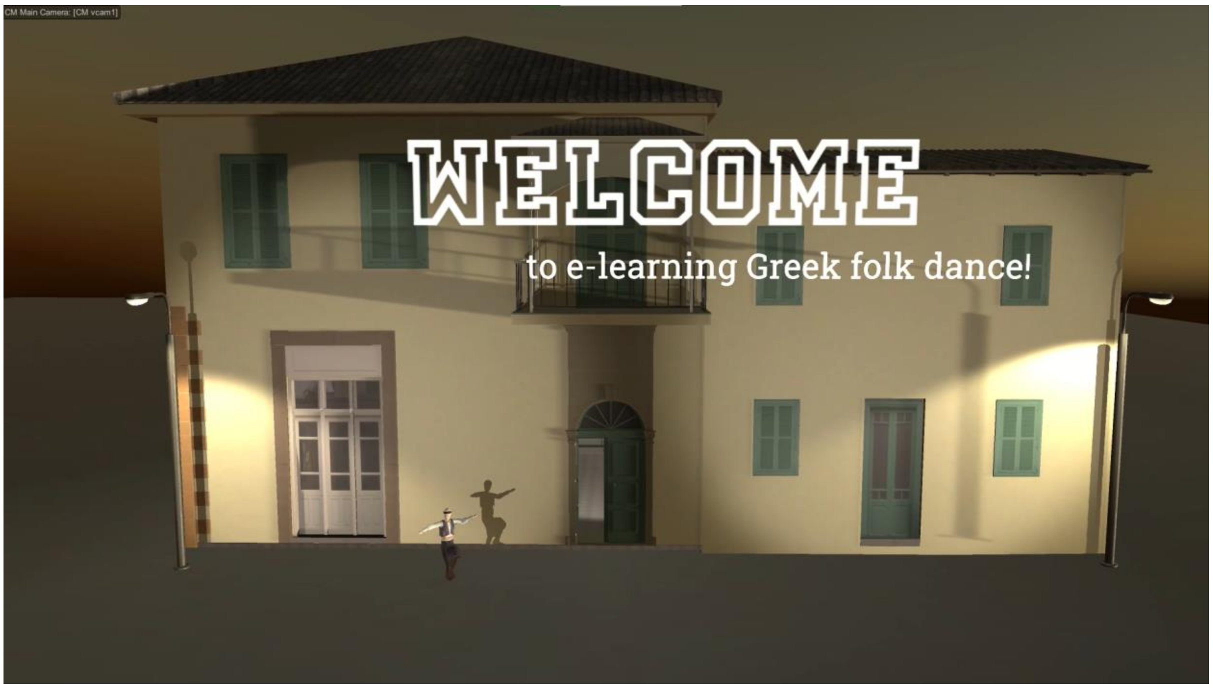


Figure 6.1.2 Gameplay Scene.

In the Gameplay Scene, the user interacts with the keyboard and mouse three times. The user must first log in if he or she already has an account or register if he or she does not. After that, either in play mode or learn mode, the user must choose a dance. Finally, the user is guided to the yard, where the gameplay starts, after selecting the dance and mode.

The mansion for this scene was given to me by Andreas Andreou. The only items I added to the scene were the light lights outside the home, which I purchased from an item in the Unity Asset store, all of the lighting sources within the home, and the floor outside the home. Apart from the objects, this scene contains all of the game's user interface, as it is the main scene with which the player will interact the most. The user interface of this platform must be straightforward, visible, and useful. In other words, the user interface for the whole game was designed in accordance with Nielsen's ten rules.

6.2 User Interface

The floating title of the game in front of the mansion, was the first thing the player observes when playing the game. The title is so large for two reasons: first, it is conspicuous, and second, the user knows the premise of the game since the title is simple and to the point.

The area of the building and their use can be seen by the figures 6.2.1 and 6.2.2. First the user is navigated on the Register/ Login area, which is at the ground floor on the left far room. After his successful log in or register he or she is navigated on the 1st floor, at the room in front of the stairs, where he or she can see the previews of the dances and decide which dance to play. Finally, the navigation ends in the dance area where, he needs to press the keyboard “space” button, to initialize Kinect as well as the button “T”, so the Tutor can begin the dance.

The navigation in these three rooms, is achieved with three cameras, that they act as a Main Camera. All of these cameras have the Cinemachine as a component. The first camera, when is activated, travels to log in/ register menu. The second camera, when is activated travels from the log in/ register menu to the selection menu. The third camera, when is activated, goes to the back yard where the game begins.

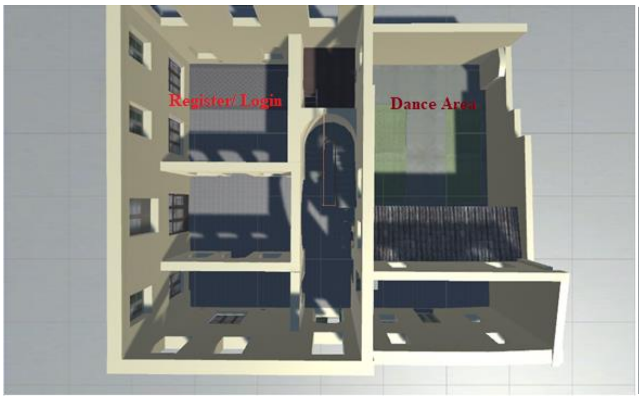


Figure 6.2.1 Ground floor and its use.

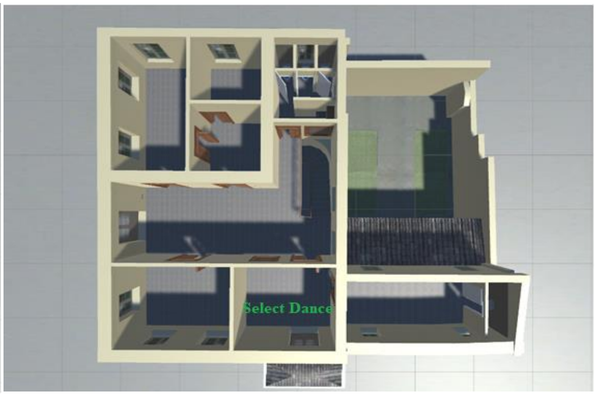


Figure 6.2.2 1st Floor and its use.

The user interface was created with a Unity Game Object named, Canvas, with a Canvas component on it. All User Interface components should be included inside the Canvas. The Canvas has a Render Mode setting, which specifies how the Canvas will be rendered to the scene. For the log in, register and selection of the dance, the render mode is World Space. For the Gameplay, the render mode is set in Screen Space. Since the platform does not support a database and it is not necessary to create one at this time, the user must use “user” as username and “1234” as password to pass the log in part, Figure 6.2.3. In case, the user enters wrong credentials, an error message is shown. The register section, does not have any functionality as it is not currently linked to a database, as seen in Figure 6.2.4.

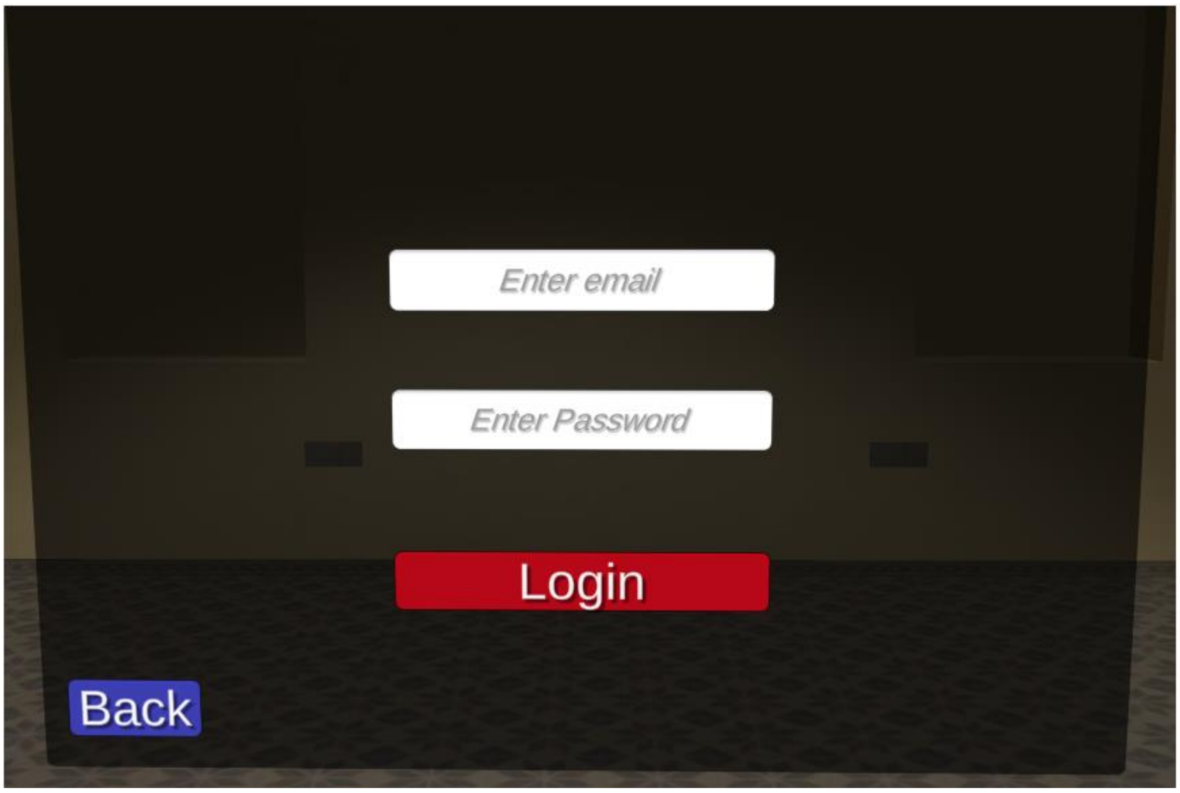


Figure 6.2.3 Screenshot of Log in Canvas.

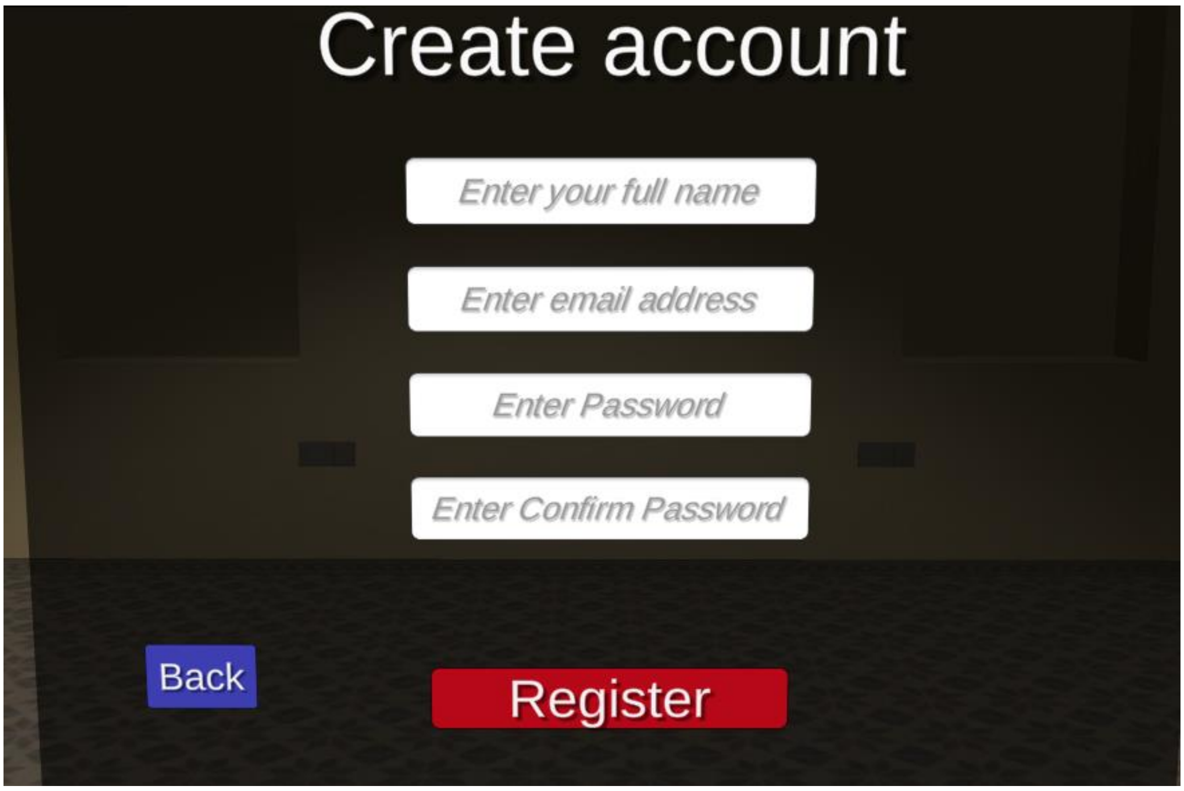


Figure 6.2.4 Screenshot of Register Canvas

For the dance selection, the platform offers five dances. By clicking on the name of the dance, the dancer shows a preview of the selected dance. Here, the user has two options “Learn Dance” and “Play Dance”. In case, the user does not select a clip and clicks on Play or Learn, a message is shown “Please select a dance first”, so he or she can understand how the Canvas here works. Again, the platform includes an error prevention system, just like the log in section, satisfying the Nielsen 10 User Interface Design Rules.

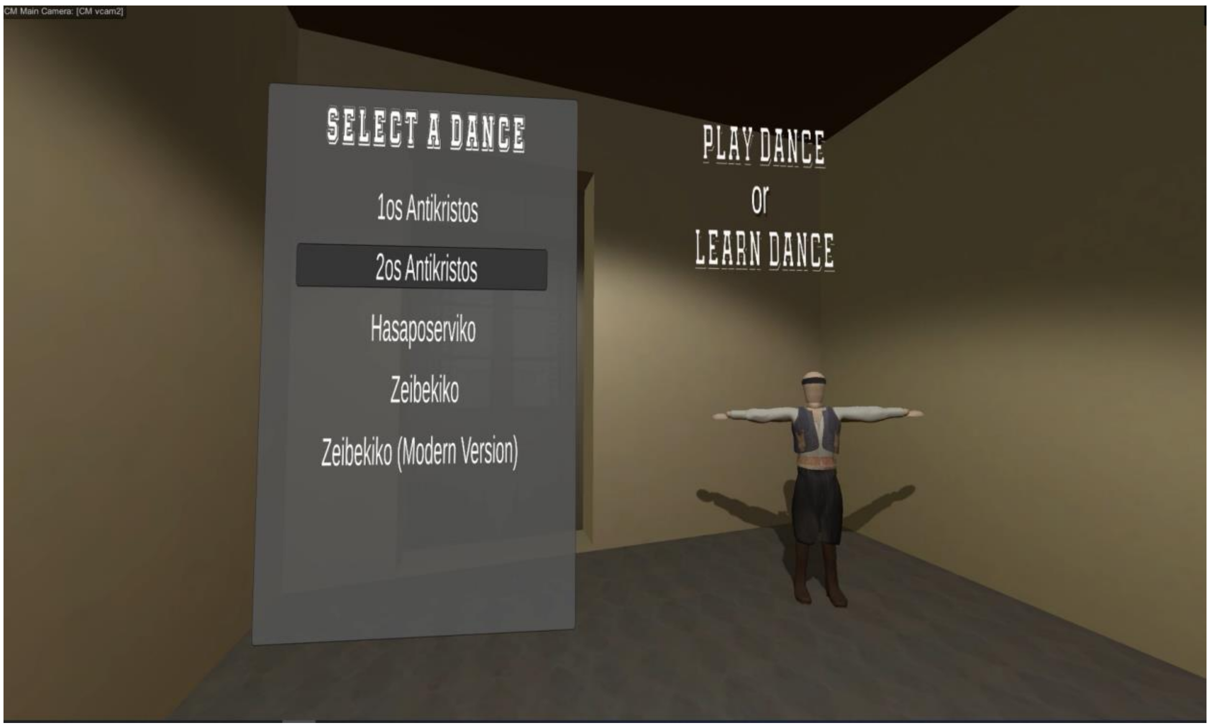


Figure 6.2.5 Screenshot of Dance selection.

After the user selected a dance, as mentioned before, the camera is navigated to the dance area. Here the Canvas, is set to Screen Space. For both modes, “Play Dance” and “Learn Dance” the gameplay Canvas is the same. The only difference is the message that it is shown above the tutor, in “Learn Dance” mode, which indicates the part of the dance that is playing.

At the end of the dance, a menu is shown, as seen from Figure 6.2.7. This menu shows the result of the user, with a message that reflects to the performance result. This message can be “Perfect dance”, “Nice Try” and “Bad Try”. The menu has also a button, named “Quit” which quits the application.

At the gameplay section, there are four characters. The main one who is the tallest of all, is the teacher. The two characters that stand in front of him, are the NPCs. The user is on the left corner of the screen, as seen from the Figure 6.2.6. To emphasize that the middle character is the teacher, I used two lights sources that pointed at him, with the same range and color, so his movement and details could be easily seen. These lights sources are set to soft shadows as shadow property.

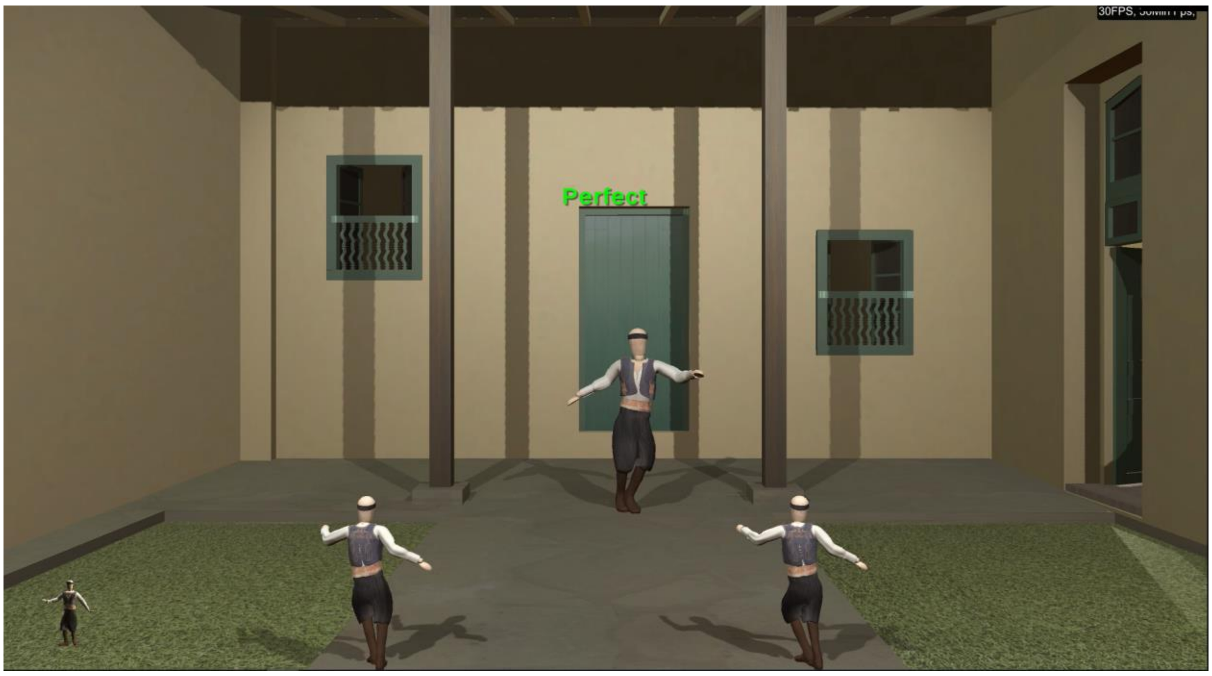


Figure 6.2.6 Screenshot of the Play Dance mode.

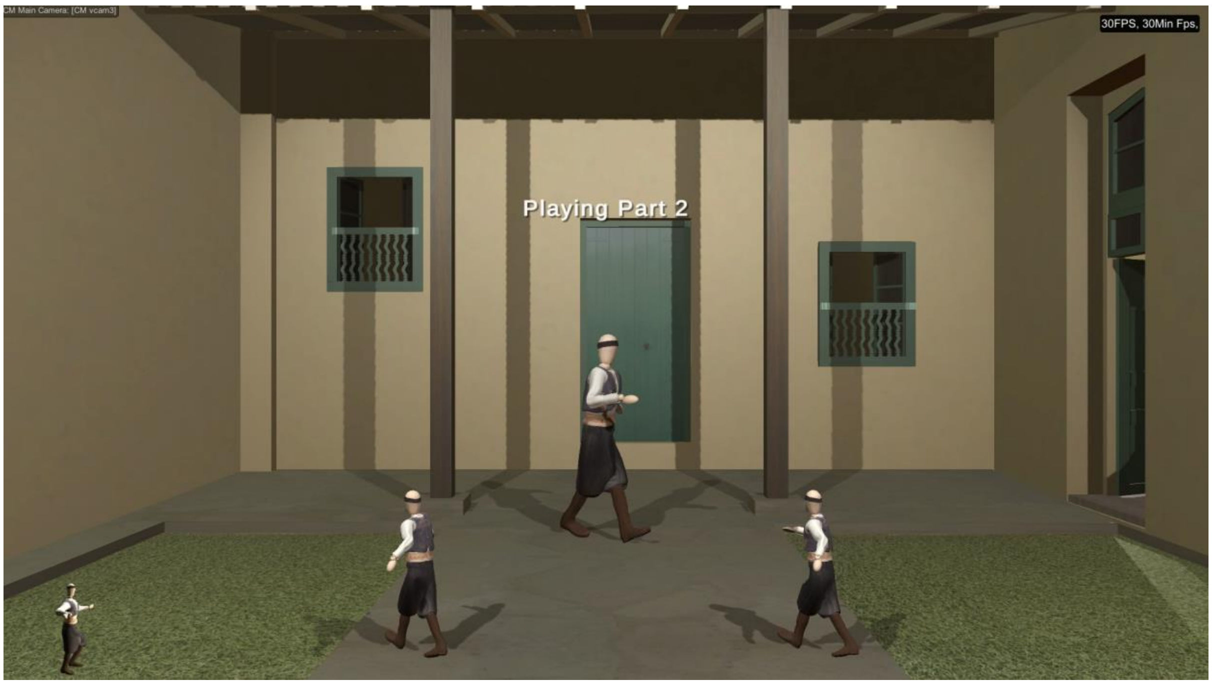


Figure 6.2.7 Screenshot of the Learn Mode.

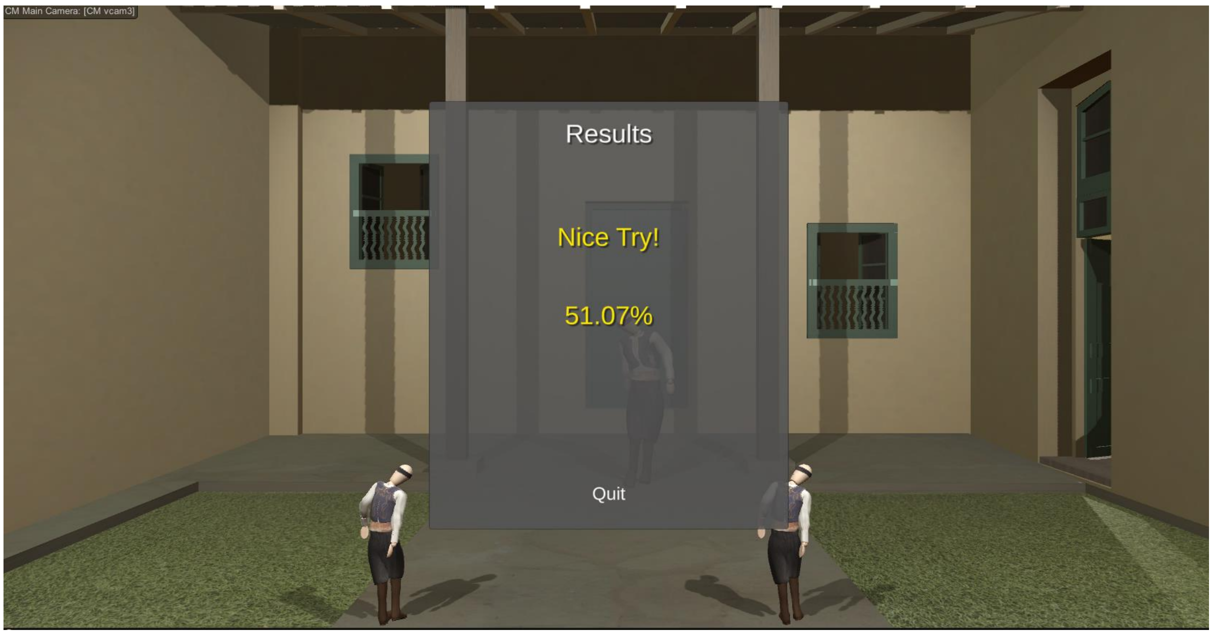


Figure 6.2.8 Final menu, showing the result of the user's performance.

6.3 Characters their creation and use

The characters of the game were created with the help of MotionBuilder, as mentioned in Chapter 3. More specific, the steps I followed to create a character was, first to download the dance from the dance database. Afterwards, having the folder of a traditional dancer characterized, all is left was to open the fbx file to MotionBuilder, the pre-captured animated dance, as seen in Figure 6.3.1, and merge it with the traditional character. This would give the result as shown in Figure 6.3.2.

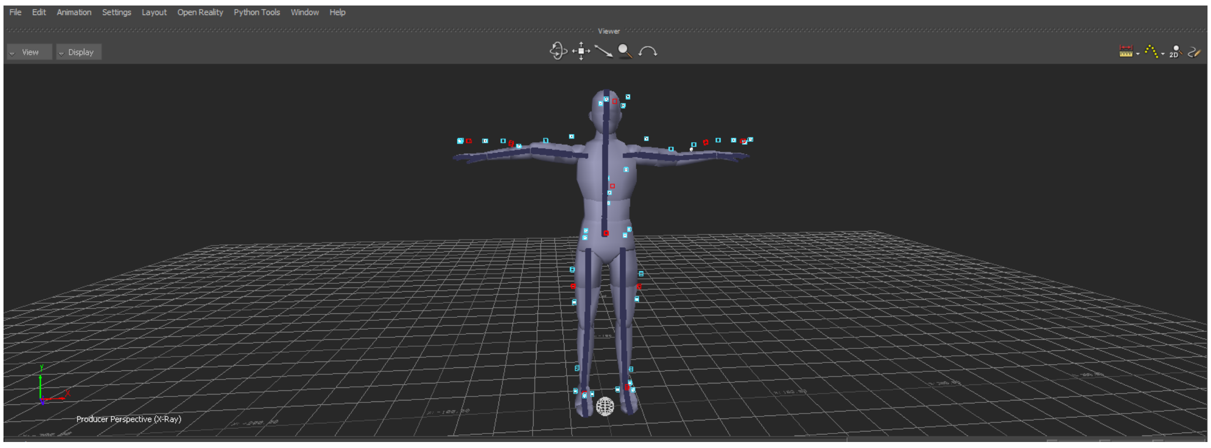


Figure 6.3.1 Motion file imported from the database.

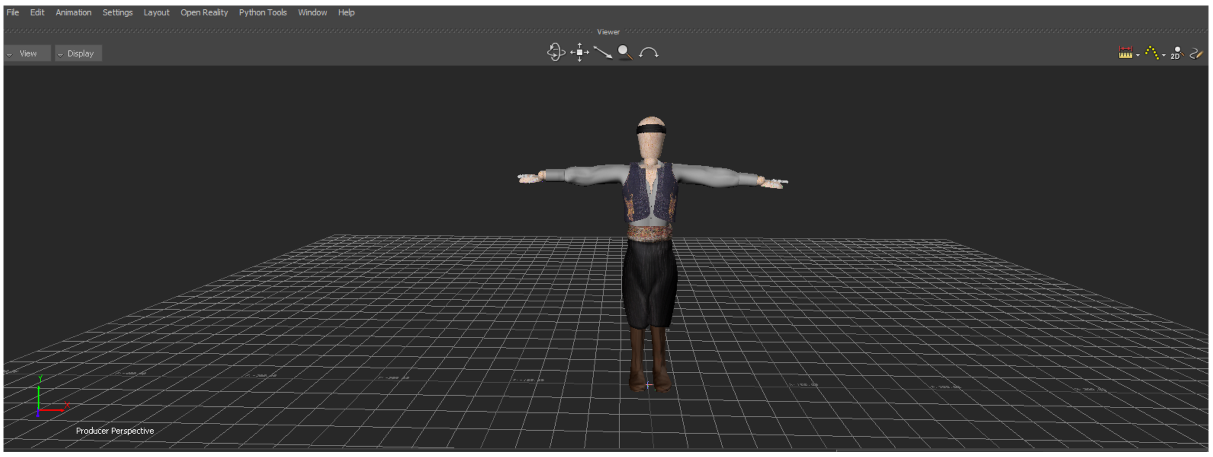


Figure 6.3.2 Merged character with motion file.

After deleting the actor, the optics, the markers and the C3D, the final part was to export the new fbx file and import it to Unity. This now is the character, who has an animation clip attached to it, the dance. All the characters have the same scale which is (1,1,1). Each character after being placed to the scene, has at least two light sources on them.

Chapter 7

Conclusion and Future Work

7.1 Review of Project

The purpose of this project is to help to improve a user's dance skills, through serious gaming. The creation of this platform really helps the user to understand the movement and the rhythm of a dance as it is user-friendly and simple.

During the implementation of the platform, I used the programming skills that I acquired from the course Object-Oriented programming CS133, as well as the course CS231 Data Structures and Algorithms, that provided me with the knowledge of many structures such as List and writing optimize algorithms. The course CS435 Human-Computer Interaction was very important as it taught me how to design a user-friendly application, by observing all the rules of good design, such as the Nielsen User Interface design guidance. In my case the application was a game. Another course that helped to understand the fundamentals of the graphics and how they work is the CS426 Computer Graphics, where I understand the principles of computer graphics, such as the translation, rotation, scale, transformation camera specification and scene graph. Needless to say, that the final project of this course, helped me understand better the Unity Engine.

I tried to solve all the problems that I encountered with, with the simple way, so the functionality of the project will keep working, even if someone soon gets the code and changes it, by removing or adding extra functionalities. The usability and the user experience of the platform I believe that are perfect, as any category of user, amateur or intermediate, can interact with it.

7.2 Scalability

The platform must have the option of scalability for a better and easier maintenance and upgrade. Documenting all of the problems, if not all of the mistakes, in this document will make subsequent production and maintenance easier.

7.3 Future Work

This platform is a game program, as has been mentioned many times. There is also room for development and extension, much as there is in every other program or application. Many modifications and enhancements can be accommodated by this interface.

The use of VR or AR environment is a move that can make the gaming more enjoyable for the player. Furthermore, expanding the Kinect capabilities, by adding the gesture listener, where the dance, for example, would begin after the user makes a particular gesture. The combination of these two ideas will make it much easier for the user, as it might no longer need the keyboard or mouse as input. The only input will be the gesture of the user and the output will be shown in the glasses. An example of this technology is the new Microsoft Mixed Reality HoloLens 2 [8].

Another potential extension that may be implemented is the development of a database that would store all the data of the users for analytics. The dances of each user performed will be stored in a table for this database, so it will be interesting to give the user the ability to import a dance that he or she finds attractive by uploading the motion file and the background music of the dance to the application. Apart from a male Cypriot tradition character, the platform may promote a female Cypriot traditional character.

Adding a new mode in which the user can learn to count tempos. Many inexperienced people, in dance, find it difficult to keep time with the beat. This mode will only play the music of a dance giving the user the opportunity to learn the tempo and rhythm of the music. This, of course, implies that the music file must be divided into segments, so the user can listen each segment individually.

Finally, in addition to recording and analyzing the user's performance, the application may also capture other aspects. Any of these aspects could include the performer's facial gestures or emotional state. Although, the evaluation could contain all the elements of LMA, as well as the synchronization of user and teacher that was mentioned in Chapter 4.2. The user will get more accurate feedback in this manner, as if they were practicing in front of a real instructor.

Many of these extensions may be used in a dance museum, where visitors could learn more about a certain dance, such as its history, while still getting the opportunity to experience it.

7.4 Conclusion

Technology is extremely important in today's world for all subjects, but particularly for educational subjects. The purpose of my bachelor thesis was to create, analyze and incorporate an educational gaming platform, that would aid users in learning and improving a specific dance of their choice, through applied graphics. Understanding how animation works, particularly in Unity, and how I can extract data from that animation was critical for me. Motion analysis is a new technique that I feel will be used in our daily lives in the future.

As a consequence, I think the platform would be extremely critical for two reasons. The first reason is, for people who have a strong desire to dance, but only have limited amount of time to do so. The second reason is that this platform may be used by individuals of all ages.

Bibliography

[1] A. Aristidou, S. Eftsathios and C. Yiorgos, “Cypriot Intangible Cultural Heritage: Digitizating Fold Dances,” 2014.

[2] Unity: http://www.unity.com

[3] A. Aristidou, “Folk Dance Evaluation Using Laban Movement Analysis,” 2015.

[4] S. Senecal, “Salsa dance learning evaluation and motion analysis in gamified virtual reality environment,” 2020.

[5] N. E. O'Connor and P. Kelly, “Evaluating a dancer's performance using Kinect-based skeleton tracking,” 2011.

[6] Dance database: http://dancedb.eu

[7] Microsoft Azure Kinect: https://docs.microsoft.com/en-us/azure/Kinect-dk/

[8] Microsoft HoloLens 2: https://www.microsoft.com/en-us/hololens