****

**Active Rehabilitation with Virtual Golf: Immersive and Virtual Player**

Name: Yong Hao, Zhou Hongyu

Ureca Project Report

**School of Computer Science and Engineering**

**Academic Year 2018/2019**

**Abstract**

The software developed with VR technology provided by HTC Vive and application of machine learning is a platform in the form of a golf game with the purposes of frozen shoulder rehabilitation.

The software will be developed with the aid of SteamVR, the Unity Engine for 3D game development, and the open-source project VRTK.

There are in total of two modes in the game. One is normal mode where the game will provide an immersive simulation of the golf game. Another is the rehabilitation mode, where user will be directed to perform certain gestures and movements for the ease of frozen shoulder in a golf game context.

1. **Introduction**
   1. **VR in rehabilitation**

Virtual Reality, by definition, is a computer technology that offers the user immersive 3-D experiences constructed on computer visuals (models etc.) and interactive elements for the users to explore. The features of the Virtual Reality can assist with various purposes, including learning and promote the rehabilitation of people with chronic joint diseases (Tania et al. 2017).

Games developed with VR technology provides rehabilitation clients from a wide range of variety with more accessible enjoyable incentive for both exercises on a full-body scale and specific practices on particular body parts (Levac et al. 2015).

Additionally, the interactions mechanisms embedded in the form of game motivating users’ continuous usage and the real-time feed-back and analysis of users’ body status and gestures, the process of the game itself distracts the user from body pain, and the increased enjoyment together, added to the popularity of VR-based rehabilitation method.

* 1. **Golf for rehabilitation**

According to a recent study conducted by Institute of Sport Science, University of Regensburg, Regensburg, Germany (2016), not only specially constructed golf training, even regular golf training poses a positive effect on the mental performance, visual spatial abilities, and physical status to the stroke patients.

Patients with chronic frozen shoulder and arm pain resulted from stroke tend to refrain from movement of arms due to the pain, which jeopardizes the recover process of the body, as clinical reports have shown regular moving practices will enhance the arm conditions.

We conclude that, the simulation of swinging a golf club in the context of a golf themed game could serve as a routinely practiced medical treatment for diseases pertaining to, or affects arm movements, such as stroke and frozen shoulder. Patients are able to follow a clinical-effective pattern of arm movements in the game.

* 1. **Approach with inverse engineering**

In the software, a selectable humanoid model reflecting the body movement of the patients will be displayed while the game is in process. The humanoid model serves as a movement reference to the patient for him/her to determine whether the current movement status fits the recreational purposes.

The method the software adopted to generate the movement of the humanoid model is by inverse engineering, which is to analyse the movements and attributes of the VR sensors in patients’ hands to generate the movement of the patient.

1. **Literature Survey**
   1. **Deep Learning with CNN**

In neural networks, Convolutional neural network is a popular deep learning architecture which is widely used to do images recognition, images classifications, objects detections and recognition faces.

CNN takes an input image, process it and classify it to be a certain category. Computer consider the image as an array of pixels, which depends on the resolution. The resolution is calculated by Height \* Width \* Dimension.

CNN is now the go-to model on the image related problems. It is successfully applied to recommender systems, natural language processing and more. CNN automatically detects the important features without any human supervision, which is the main advantage of CNN compared to its predecessors. For example, given many pictures of cats and dogs it learns distinctive features for each class by itself.

CNN is also computationally efficient. CNN uses special operations like convolution and pooling. It also performs parameter sharing. CNN can run on any devices, making them universally attractive,

* 1. **Inverse Kinematics**

[Add a section …]

1. **Methodology**
   1. **VR system of Unity**

Unity VR helps you to target VR devices directly from Unity, with none external plug-ins in projects. There are a base API and have sets with compatibility for multiple devices. it's been designed to supply forward compatibility for future devices and software system.

* 1. **SteamVR**

SteamVR is a video game system with a receiver known as HTC Vive that is constructed by Valve and partner HTC. just like the Oculus Rift, the Vive is a VR device that contains 2 screens (1080x1200 resolution) streaming data at high refresh rates (90Hz) to make the sense of virtual reality. It uses lenses that sit between user’s eyes and the displays to make a rounded field of view and facilitate them to concentrate on the LCDs. The receiver can connect via a hardwire to a computer, which is able to run games and alternative VR software.

There is another component that makes up SteamVR is a pair of laser-emitting base station known as lighthouse. The two base stations are placed within the corners of the area to map the boundaries of a space. The last component is a pair of controllers with sensors. There are several buttons to manage the actions and directions.

* 1. **VRTK**

VRTK is an assortment of helpful scripts and ideas to help building VR solutions quickly and easily in Unity3d 5+.

* 1. **Humanoid Avatar**

Unity’s Animation System has special characteristics for operating with humanoid characters. Unity provides a specialised workflow, associated an extended tool set for humanoid animations because of the common humanoid characters in the game.

The Avatar system told us how Unity spot that a specific animated model is humanoid in layout, and that elements of the model correspond to the legs, arms, head and body.

Because of the similarity in bone structure between completely different humanoid characters, it's attainable to map animations from one humanoid character to a different, permitting retargeting and inverse kinematics(IK).

* 1. **Animation in Unity**

Workflow and setup for animation for all constitutive of Unity including objects, characters and properties become easily in Unity. For the Animation system, it supports to import animation clips and create animation within Unity. Also, it is able to apply animation from one-character model into another. The preview of animation clips is convenient, which makes animators and programmers work more independently.

Unity has an abundant and complicated animation system, which is based on the concepts of animation clips.

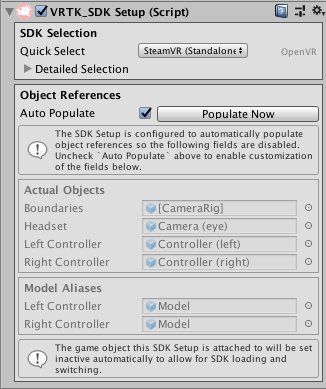
1. **VRTK Fundamentals**
   1. **Mechanisms**

VRTK is a collection of SDK which makes it more convenient for user to develop project with less coding. Using only SteamVR and Unity with the scripts that they provide also need users to do a lot of programming. VRTK provides scripts which can be easily added to the component of objects. It can be imported from assert store or Github which depends on the version the user needs.

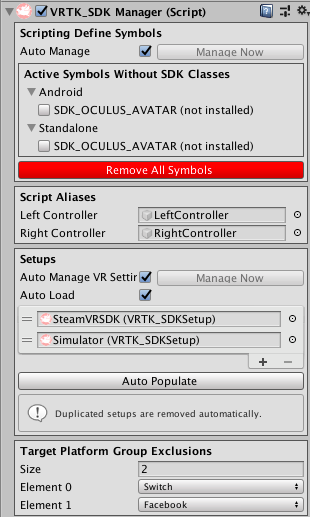
* 1. **SteamVR SDK setup**

Steam VR plugin will be downloaded and imported from assert store. after importing finished, we can use a lot of scripts and prefabs that SteamVR, VRTK and Unity provide. The CameraRig prefab from SteamVR will be used as the main camera.

First, build a new empty object named VRTKSDK, inside the VRTKSDK, build a new empty object named SteamVRSDK and attach CameraRig prefab and SteamVR prefab to it. After do this, add VRTK\_SDK Setup script to be component of SteamVRSDK.



In the quick select filter, choose SteamVR. And then, for the VRTKSDK, need to select SteamVRSDK and Simulator in the setups part and then Auto Populate.

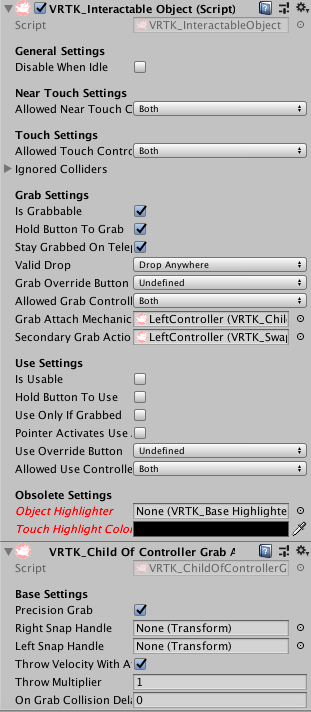


* 1. **Controllers setup**

Create a new empty object named VRTKScript. Under the VRTKScript, create a couple of empty objects, named them as LeftController and RightController.

In VRTKSDK, we set the Script Aliases to be these two controllers.

Under the two controller objects, add VRTK\_Pointer Script which achieves the functionality to do nicer teleporting than straight pointer. Then, add the pointer rendering to finish the setup of pointer. After that, we add VRTK\_Controller Event to both controllers, which will be helpful for teleporting. VRTK\_Interact Touch and VRTK\_Interact Grab will be added to controllers to achieve the goal to grab objects. We need to add VRTK\_Interactable Object and VRTK\_Child Of Controller to finish the setup of controllers.



1. **Game Procedures**
   1. **Basic Gaming Functions**

The game itself works as a golf game to serve the purpose of rehabilitation of the player. We assume the player are patients suffering from shoulder or movement-related diseases, such as frozen shoulder

* 1. **Entering the game**

Upon entering the game, the player will start off at the center of the terrain. At this time the player would observe the surrounding massifs and the plantations on the border. The player would observe a parasol nearby, and a framework upon which different types of golf wedges lie (At this development stage, only one particular wedge is available for interactions with the golf ball).

* 1. **Game Interactions**
     1. **Player with the wedge**

The player could grab the wedge by contacting the right controller with the wedge and press the index trigger. While on contact, the wedge’s material will be changed to deep green. And while grabbing the wedge, the wedge’s material will be changed to deep pink. By pressing the hand trigger, the player is able to rotate the wedge on hand. A degree of pi/6’s rotation will incur for each time the trigger is pressed.

* + 1. **Wedge with the golf ball**

The script attached to the golf ball will detect the collision with the golf wedge only. Upon contacting, the golf ball will be assigned a initial velocity same as the golf wedge’s velocity to mimic the strike of the golf wedge. The velocity might be multiplied by a changeable multiplier to control the strength of the strike.

* + 1. **Golf ball**

The golf ball applied trail renderer to trace the movement trail of the golf ball. The purpose of the trail renderer is for the player to trace the golf ball more easily.

* + 1. **Golf ball with the golf ball hole**

The script attached to the golf ball hole will detect the collision with the golf ball. Upon collision, the golf ball game object will be destroyed, and the player will gain one score.

* + 1. **Director**

A golf wedge is designed to point to the direction of the golf ball hole. The purpose for this direction is because the large range of the terrain area makes it difficult to locate the golf ball hole.

1. **Implementation** 
   1. **Game environment – Terrain**
      1. **Terrain Overview**

To build an immersive gaming environment of a golf field and to enhance the nourishment effects the flourishing environment it can produce upon users, the game adopted following techniques to achieve the goals: Unity Terrain Engine was adopted to build the initial terrain. Two kinds of different grass material were applied to the terrain built to mimic the real-life golf fields. As for the background music, we have mixed the bird sounds and the sound of winds together and played them in the background on default. Research has been conducted on the natural soundwaves could incur positive effects on people’s physical and mental recoveries (source).

* + 1. **Unity Terrain Engine**

Unity provides a primitive terrain system that allows the user to create and modify landscapes into the game scene. The advantages of such a built-in system are summarized as following: performance, convenience and modifiability.

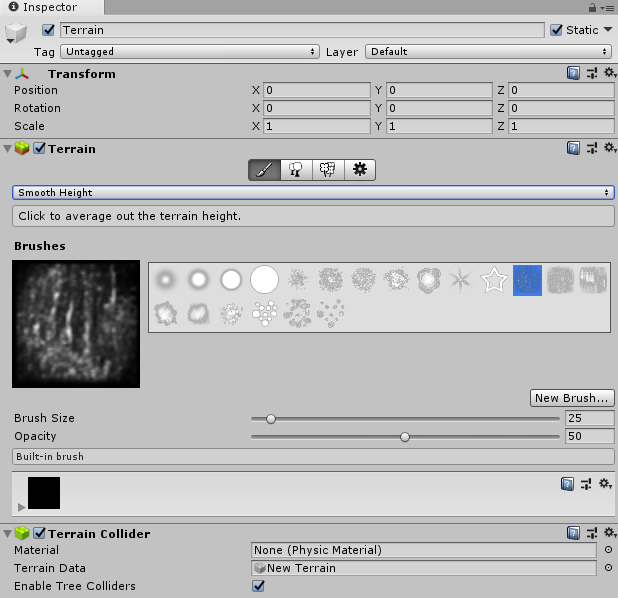


Figure. The editing tools of terrain

**Performance:** The rendering system designed for terrain are highly optimized. The system includes an additional GPU-instanced render path for terrain since the March update on terrain system, resulting in dramatic reductions in the number of draw calls issued, leading to an average reduction of more than 50% of render time compared to previous versions.

**Convenience:** The unity terrain system provides multiple tools to add detailed features to the terrain. The functions they provided includes raise and lower the terrain heights, adding trees and other details to the terrain, smoothing the terrain etc. The changes made to the terrain are rendered real-time, so that the results are reflected visually to the users.

**Modifiability:** Using the tools provided are simple for a non-terrain artist, and similar to using brushes and paint on the canvas. The brush here defines the multitude and area on which you would like to change for the terrain. As mentioned above, two intersecting grass materials were applied to the terrain to better mimic real-life golf sports field. Multiple types of trees are added to the borders of the terrain as well.

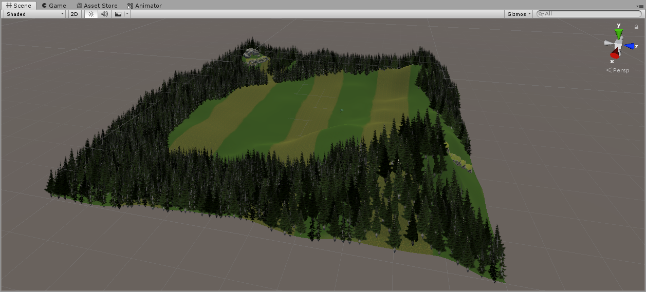
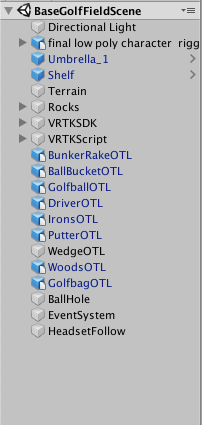


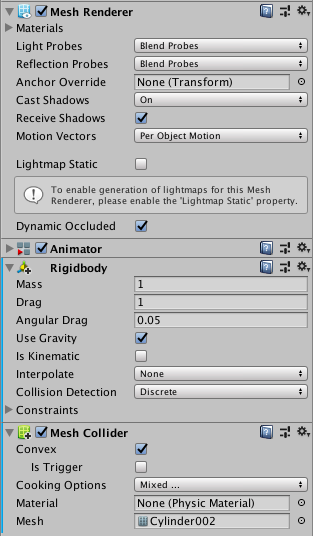
Figure Terrain in game

* 1. **GameObjects**

GameObjects are the fundamental objects in Unity that represent characters, props and scenery. They do not accomplish much in themselves, but they act as containers for Components, which implement the real functionality. Every GameObjects will have a component named Transform which is used to represent position and orientation, and it cannot be removed. The other components that implement its functionality can be added from the editor’s component menu or from a script. There are some primitive objects that can be added from GameObject-> 3D Object menu.



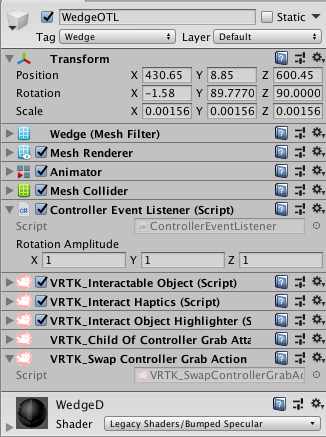
1. Directional light is the light source of our Golf VR game.
2. The Avatar is built to simulate user’s actions, which is the base of Inverse Kinematics.
3. Terrain is the base game environment of a golf field. The details of terrain can be found in part 6.1.
4. Umbrella, BunkerRankeOTL, DriverOTL, IronsOTL, PutterOTL, GolfbagOTL, Rocks and BallBucketOTL are all the decorations in the game. They need to have Rigidbody and Mesh Collider to prevent that they will fall across the terrain. For the Umbrella, we add a Audio Source as its component, so that when user start get into this game, they will hear the sound of wind and bird.



1. The shelf is used to put kind of golf equipment.
2. VRTKSDK is used to set up SteamVRSDK and the VRTKScript is used to set up controllers.
3. WedgeOTL is used to hit the golf ball. It should be interactable. User needs to grab WedgeOTL and use it collide with golf ball to hit it to the destination. After user picks up the wedge, user can rotate the wedge to have a better direction to hit the ball by press the buttons on the two sides of controllers.

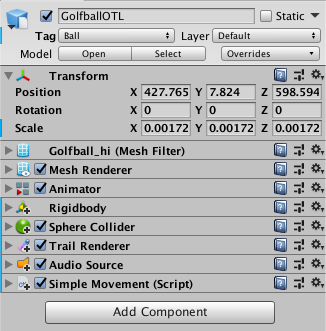
The Script is showed below:

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
using VRTK;  
public class ControllerEventListener : MonoBehaviour {  
    public Vector3 rotationAmplitude;  
    private GameObject wedge;  
  
    // Use this for initialization  
    void Start () {  
        GetComponent<VRTK\_ControllerEvents>().GripPressed += new ControllerInteractionEventHandler(RotateGolfClub);  
    }  
    // Update is called once per frame  
    void Update () {}  
    void RotateGolfClub(object sender, ControllerInteractionEventArgs e)  
    {  
        wedge = GetComponent<VRTK\_InteractGrab>().GetGrabbedObject();  
        Vector3 targetRotation = wedge.transform.rotation.eulerAngles + rotationAmplitude;  
        wedge.transform.rotation = Quaternion.Euler(targetRotation);  
        Debug.Log("Grip is pressed and target rotation is " + targetRotation);  
    }  
}



1. GolfballOTL will have movement if the ball is collided with WedgeOTL. This collision will have a hit sound due to the Audio Source component.

The movement is achieved by Simple movement Script. In the Script, golf ball will have a velocity according to the wedgeVelocity after they have collision. The wedgeVelocity can be calculated according to its positions and time. If the golf ball is moving, it will be added a force whose direction is opposite to the direction of movement.

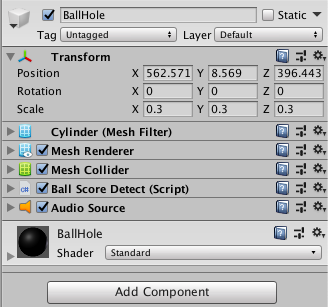


using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class SimpleMovement : MonoBehaviour {  
    public GameObject wedge;  
    public Vector3 wedgeVelocity;  
    public Vector3 heightAdjustment;  
    private Transform wedgeTransform;  
    private Rigidbody ballRigidbody;  
    private Vector3 lastPosition;  
    private Vector3 currentPosition;  
    private TrailRenderer ballTrailRenderer;  
    // Use this for initialization  
    void Start () {  
        ballRigidbody = GetComponent<Rigidbody>();  
        lastPosition = wedge.transform.position;  
        ballTrailRenderer = GetComponent<TrailRenderer>();  
        ballTrailRenderer.enabled = false; }  
    // Update is called once per frame  
    void Update () {  
        currentPosition = wedge.transform.position;  
        wedgeVelocity = (currentPosition - lastPosition) / Time.deltaTime;  
        lastPosition = currentPosition;  
        Fraction();  
    }  
    void Fraction()  
    {  
        if(ballRigidbody.velocity.magnitude != 0)  
        {  
            ballRigidbody.AddForce(new Vector3(ballRigidbody.velocity.x, 0, ballRigidbody.velocity.z) \* -0.5f);  
        }  
    }  
    private void OnCollisionEnter(Collision collision)  
    {  
        Debug.Log("The ball has been in contact with " + collision.collider.tag);  
        if (collision.collider.tag == "Wedge")  
        {  
            ballRigidbody.velocity = wedgeVelocity \* 20 + heightAdjustment \* wedgeVelocity.magnitude;  
            ballTrailRenderer.enabled = true;  
            AudioSource ballSwingAudio;  
            ballSwingAudio = GetComponent<AudioSource>();  
            ballSwingAudio.Play(); }}}

1. BallHole is a Cylinder which only has small part above the terrain. If the golf ball and this cylinder have collision, the user get the score. The script is showed below:

using System.Collections;  
using System.Collections.Generic;  
using UnityEngine;  
  
public class BallScoreDetect : MonoBehaviour {  
    public GameObject golfBall;  
    public Vector3 origialPos;  
    private AudioSource scoreAudio;  
    // Use this for initialization  
    void Start () {  
        scoreAudio = GetComponent<AudioSource>();}  
    // Update is called once per frame  
    void Update () {}  
    private void OnCollisionEnter(Collision collision)  
    {  
        if(collision.collider.tag == "Ball")  
        {  
            Debug.Log("The ball has been scored");  
            scoreAudio.Play();  
            Instantiate(golfBall, origialPos, Quaternion.identity);  
            Destroy(golfBall);  
        } }}





* 1. **Inverse Kinematics**
     1. **Principles of IK**

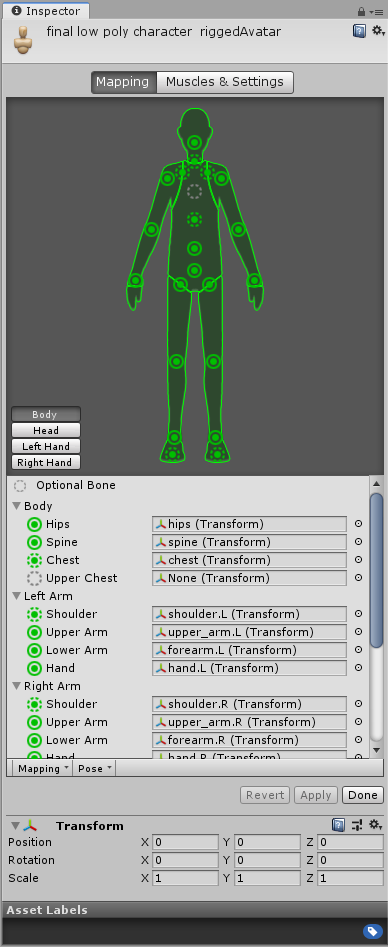
Animations are created usually by rotating the joints of a skeleton model by a certain angle. Along with the parent joints, child joints will rotate accordingly, thus to predict the final position of a series of joints, it is sufficient to derive the result from the angles and relative positions of the individual joints it contains. Modifying a skeleton model this way is defined as “forward kinematics”.

However, forward kinematics can cause very complex calculations to determine the rotations and conditions of the joints in between, given a designated position for the end joint. Sometimes, for example, it’s relatively easy to determine the landing position of the feet or the hand, but hard to determine the joints from leg to ankle or from shoulder to palm using forward kinematics.

“Inverse Kinematics” provided a solution to this problem. Given a chosen position for the end point of a series of joints, inverse kinematics works backwards to determine a possible way of rotating cascading joints so that the end point lands where we want it to land.

* + 1. **Avatar Setup and Model-Rig Mapping**

To use the inverse kinematics of unity, the desired model to modify must have a validly setup avatar and the rigs are mapped accordingly, especially for humanoid models. Unity reconciles the bone structure of the model with its animation whenever it’s imported. Unity achieves this by mapping some bones in the model file to a humanoid avatar in order to ensure the animation works properly.



* + 1. **Inverse Kinematics Design Goals**

Since the player of the game are most times using first-person perspective to observe the game scene, it is essential for the player himself/herself to be able to observe his/her body movement as well, to modify his/her gestures according to his/her rehabilitation objectives. Additionally, it is also important for the game system to determine the desirable gestures for the players to pose and warn the players if the gestures they’re posing might potentially cause harm to the players.

* 1. **Grab object with both hand**

1. **Conclusions**

**References**

1. Crocetta, T. B., Araújo, L. V., Guarnieri, R., Massetti, T., Ferreira, F. H., Abreu, L. C., & Monteiro, C. B. (2017). Virtual reality software package for implementing motor learning and rehabilitation experiments. *Virtual Reality,22*(3), 199-209. doi:10.1007/s10055-017-0323-2
2. Levac D, Espy D, Fox E, Pradhan S, Deutsch JE (2015) “Kinect-ing”  
   with clinicians: a knowledge translation resource to support decision making about video game use in rehabilitation. Phys Ther  
   95(3):426–440. doi:10.2522/ptj.20130618
3. Petra J., Tobias S. (2016) T*he* *Improvement of visual-spatial performance after golf training in patients with stroke: a pilot Study.* Institute of Sport Science, University of Regensburg, Regensburg, Germany. ASCR Publishing House. ISSN: 1224-8398. Volume XX, No. 3 (September), 159-169.
4. Technologies, U. (n.d.). Terrain Engine. Retrieved from https://docs.unity3d.com/Manual/script-Terrain.html
5. Technologies, U. (n.d.). Importing humanoid animations. Retrieved from https://docs.unity3d.com/Manual/ConfiguringtheAvatar.html
6. Technologies, U. (n.d.). Inverse Kinematics. Retrieved from https://docs.unity3d.com/Manual/InverseKinematics.html