Abstract: Methods and computer programs that allow users to move game objects in a virtual space with movements performed on current and future portable devices. By using existing technologies in such devices, the methods compute movements relative to the devices themselves in order to detect swinging motions, panning motions, moving motions and gestures and then to apply them to game objects and game cameras inside the virtual space. Moving the physical device causes the devices sensors to respond. The methods interpret this data and provide real time updates within a virtual space on the device as well as transmit the updates to another device that is also moving. This enables 2 players to connect to each other and each player is able to move their device and see it reflected within the others virtual space. This eliminates the need of real world references.
MOTION DETECTION FOR PORTABLE DEVICES

BACKGROUND


Field of the Invention

[02] The present invention relates to methods and computer programs to detect motion of a portable device and translate it to gestures to be used in game objects in the virtual space in single and multiplayer sessions.

Description of the Related Art

[03] Virtual Reality (VR) is a computer-simulated environment, whether that environment is a simulation of the real world or an imaginary world, where users can interact with a virtual environment or virtual artifact either through the use of standard input devices or specialized multidirectional devices. The simulated environment can be similar to the real world, for example, simulations for flight and racing conditions, or it can differ significantly from reality, as in VR games. Virtual reality is often used to describe a wide variety of applications commonly associated with its immersive, highly visual three-dimensional (3D) environments. The development of Computer Aided Design (CAD) software, graphics hardware acceleration, head mounted displays, database gloves, and miniaturization have helped popularize the notion.

[04] Artificial Intelligence (AI) is a series of algorithms that perform tasks in a similar way a human would without any interaction from a real human factor besides feedback. These
algorithms although limited to one specific functionality, can even outperform humans and adapt to new conditions faster in any real or virtual environment.

Portable Device is a generic name that encapsulates general purpose devices that can be wearable such as smart watches and the ones can be carried comfortably by the user like smart phones, tablets and mp4 players with Android or iOS systems, and dedicated devices. The portable devices mentioned contain three accelerometers and a three dimensional gyroscope for motion controls. Head facing cameras in these devices are optional but covered as well.

SUMMARY OF THE INVENTION

The present invention is directed to a system for detecting motion of a portable device and the interpretation of the motion for use in controlling 3D virtual objects within a 3D virtual world as generally used within games. The motions which are interpreted are swinging, panning and moving. Gestures which are triggered to perform specific in game actions are also included. This is accomplished through the use of a linear algorithm, a neural network algorithm, a fuzzy logic algorithm, and a face tracking algorithm. The calibration and transmission of these interpreted motions to another portable device is also provided.

Embodiments of the preset Invention provide methods for controlling virtual objects using a mobile phone device that contains at least accelerometers, gyroscope, CPU and data storage device. The methods described herein detect specific movements of the device by a user's motion of the device. These movements are interpreted by algorithms which are described in these methods. The algorithms which detect the motion of the device are separated into hitting motions, panning motions and gesture motions. These enable the user to
perform operations such as throw and catch, throw and hit and hit and hit of an object such as a ball. The following general algorithms are utilized and adapted into the specific methods and algorithms used for the above stated purposes. These generic algorithms are the linear method, neural network method, fuzzy logic method, facial methods (face tracking, face calibration, and face transmission to a game object such as a paddle or glove or other game object) and video camera tracking method. Finally, this patent describes a method for connecting 2 or more of the 3D worlds together and the method of calibrating the mutually shared game experience. This enables interaction within a shared 3D world to occur and allows the same experiences to be transmitted to all participators without the need for the participators to be positioned in the same way.

[08] Three hardware devices that are used for the hit detection defined in these methods are accelerometers, gyroscopes and a front-facing camera.

[09] The methods describe the use of the movements of an user to position a game object within a 3D world. The game object may be a camera, or a paddle or the camera and the paddle. Some of the methods describe ways of advanced control of virtual objects in the 3D world using a real, common portable device that exists in the market and performing actual movements with said device. The primary purpose of connecting a game object such as a paddle to the user is to allow the user to interact within the 3D world as though playing racket sports which require hitting or throwing and/or catching of a ball. Thus particular emphasis and focus is employed in the algorithm for detecting a user's hitting motion and applying this motion within the 3D world to enable the user to experience hitting a ball in the 3D world with their device. The methods described herein for determining the hitting motion from the users
movements have been designed according to the following 5 algorithms: linear, neural, fuzzy, face referencing, and camera referencing. (Illustration of movement made by a user moving a device being reflected in the virtual world's as a paddle with the camera following it).

**BRIEF DESCRIPTION OF THE DRAWINGS**

[010] The inventions may be best understood by reference to the following description taken in conjunction with the accompanying drawings in which:

[011] FIG. 1 Describes a common knowledge of the three linear and rotational axes present in any common portable device that exists in the market.

[012] FIG. 2 Illustrates the representation of the real world device with a virtual object that can either be seen by the game camera or be the game camera itself.

[013] FIG. 3 Illustrates how the linear movement of the portable device will be reflected on the game object in the 3D world using different methods that will be described later.

[014] FIG. 4 Shows the flowchart of the process of using the linear acceleration data to detect the start or end of a linear movement in the virtual world.

[015] FIG. 5 Contains the flowchart of the use of head tracking data to change position and rotation of game objects or game camera.

[016] FIG. 6 Shows the flowchart that explains how to use head tracking data to calibrate and train a neural network that users accelerometers.

[017] FIG. 7 Explains the process of detecting a hit motion based on linear or rotational changes on a portable device.

[018] FIG. 8 Illustrates how the rotational movement of the portable device will be reflected on the game object in the 3D world using different methods.
FIG. 9 describes the process of detecting rotation tricks using captured data.

FIG. 10 illustrates a way to change the rotational effects of the game world by changing the pivot point to any desired location related to the game object to be controlled.

FIG. 11 illustrates how the local motion coordinates change depending on the orientation of the portable device and it can be detected and used with different methods.

FIG. 12 illustrates the method of re-centering the game’s camera to any orientation the user desires to play while keeping the playing ground ready for actions.

FIG. 13 shows a case of multiplayer game session using a virtual reference point without the need to use a physical reference point to synchronize the data between players like in a game environment such as table tennis.

DETAILED DESCRIPTION

The following embodiments describe methods and computer programs for existing portable devices that provide ways to detect motions caused by a user and interpret them to control virtual objects in a game scene in a way that detects a human natural movement. The methods contain programs that offer a different approach to obtain similar results but are independent of one another, this means that any method or program can be used and it will depend on the scope of the application or game that a program will be best suited for. The hardware of technology present in the current portable devices have not been described in detail due to general knowledge of them and to not obscure the present invention.

FIG. 1 Describes a common knowledge of the three linear and rotational axes present in any common portable device that exists in the market. These axes are y or yaw, x or pitch.
and z or roll axes for linear or rotational movement respectively.

[026] FIG. 2 Illustrates the representation of the portable device shown as 202 with a virtual object that can either be seen by the game camera like in 203 or be the game camera itself as in 204. The invention allows to adapt to any app design situation and fit development needs for motion controlled object in the virtual world. Any change in the device's orientation will be represented in the virtual work and any changes in position of rotation can be used to provide more inputs than what a touch screen can provide for natural use.

[027] FIG. 3 302 shows a typical example of changes in position of the portable device like in 303 made by the user and these changes are reflected on the virtual object 304 that will move in the corresponding axis. It does not detect the position, but changes that are easily found thanks to the information provided by the accelerometers and gyroscope and interpreted by the computer program. The changes in position are not only used to move objects in the 3D space but also to detect a motions such as hits with a paddle, racket or bat and calculate a force value in any of the axes. The programs that perform these tasks can be AI algorithms like fuzzy logic and neural networks for precision, head tracking algorithms that use the player's head as reference point to perform movements in the 3D space where precision is a must but not performance in the simulation, or simple linear calculations for quicker performance at expenses of accuracy. It depends on the application or game design requirements which one will be best used.

[028] FIG. 4 Illustrates the process in which the linear accelerations is processed by computer programs to detect the start and end motions in the captured raw data. When the device is not moving at the beginning, it will wait for a start signal in the captured data using either linear
calculations, fuzzy logic or neural networks and if so, it will calculate the speed in which the game camera or object will move and will remain in that state until a stop signal is detected using the same computer programs.

[029] A class defined as Capturer.cs (a C# script) takes care of the capturing of the accelerometer, gyroscope and face tracking (if supported on device) data where and puts it in a structure like the following:

[030]

```
public struct AccelerometerData
{
    public float timeframe;
    public Vector3 rawDDataValue;
    ...
}
```

[031] This data structure will allow the algorithm to see the changes of the captured data over time for detection of any motion that is desired to detect, including the hit detection. Every time the data is captured it is stored in a collection that is accessed by the evaluation code.

[032]

```
public List<AccelerometerData> rawGyroAccelData = new List<AccelerometerData>();
public List<AccelerometerData> rawGyroRotData = new List<AccelerometerData>();
public List<AccelerometerData> rawHeadTrackDataPos = new List<AccelerometerData>();
public List<AccelerometerData> rawHeadTrackDataRot = new List<AccelerometerData>();
```

[033] The list rawGyroRotData is created to store the value of the rotation rate of the device (similar to degrees per second but not really accurate units - [because it is storing the items as the game engine retrieves the raw data and this data is not consistent thus we must processes this data based upon the devices processing power and record the time unit, then when applying the algorithm upon the data the time metric defines the degree of accuracy]) and
rawGyroAccelData is to store the linear acceleration data (similar to meters per second squared but again, not really accurate units).

[034] The Capturer Class, after obtaining the accelerometer data, stores it in the first element of the rawGyroAccelData list. This way the most recent data is always on the first elements of the list.

[035]

rawGyroAccelData.Insert(new AccelerometerData(0, currentTime, Input.gyro.userAcceleration));
rawGyroRotData.Insert(new AccelerometerData(0, currentTime,
                             Input.gyro.rotationRateUnbiased));
rawHeadTrackDataPos.Insert(new AccelerometerData(0, currentTime,
                           HeadTracker.position));
rawHeadTrackDataRot.Insert(new AccelerometerData(0, currentTime,
                            HeadTracker.rotation));

[036] On the snippet above, currentTime is the time the data has been captured and Input.gyro.userAcceleration along with Input.gyro.rotationRateUnbiased, HeadTracker.position and HeadTracker.rotation corresponds to the linear acceleration date, rotation, and head tracking data respectively detected at that time. To avoid memory blowout, a number of sample's parameters are defined so that when the lists achieve the full capacity determined by such parameter, it will discard the last element of the list when a new element is inserted at the beginning to keep the memory use constant. There is also another parameter, a sampling time, in which each data is stored after a certain time has passed (from millisecond to seconds depending on the requirements of the game). This is similar to a buffer of data.

[037] For the linear movements, it is necessary to have at least two values to compare, but three are recommended in order to detect a peak or valley value that will determine the
direction and speed of the virtual object to move and even a force value if the data is used to
detect a hit motion for a paddle, racket or bat.

[038] The code for detecting a linear motion and stopping it can go as follows.

[039]

```
Vector3 data1;
Vector3 data2;
Vector3 data3;
Vector3 isMovingVector;
float moveSpeed;
...

data1 = Capturer.rawGyroAccelData[0].rawDataValue;
data2 = Capturer.rawGyroAccelData[1].rawDataValue;
data3 = Capturer.rawGyroAccelData[2].rawDataValue;
...

transform.position = transform.position + (isMovingVector * moveSpeed);

if(isMovingVector.x == 0) // process movement in x axis
{
    if(data2.x > data1.x && data2.x > data3.x) //peak
    {
        isMovingVector.x = Mathf.Clamp(data2.x, -If If); // Movement in opposite direction
    }
    else if(data2.x < data1.x && data2.x < data3.x) //valley
    {
        isMovingVector.x = Mathf.Clamp(data2.x, -If If); // Movement in opposite direction
    }
}
else
{

```
If(isMovingVector.x > 0 & & data2.x < datal.x & & data2.x < data3.x) // stop signal
isMovingVector.x = 0;
If(isMovingVector.x < 0 & & data2.x > datal.x & & data2.x > data3.x) // stop signal
isMovingVector.x = 0;
}
// Do the same for "y" and "z" axis

[040] AI algorithms like Neural Networks and Fuzzy Logic can add a better level of precision using three separate systems (one for each axis) to interpret the motion data by the accelerometers and gyroscopes in linear acceleration. The following algorithm shows how to use Neural Networks and Fuzzy logic to determine the panning movements of the user:

[041]

Vector3 resultVect;
Vector3 maxValueInputVect;
Vector3 maxValueOutputVect;

Vector3 datal = Capturer.rawGyroAccelData[0].rawDataValue;
Vector3 data2 = Capturer.rawGyroAccelData[1].rawDataValue;
Vector3 data3 = Capturer.rawGyroAccelData[2].rawDataValue;

if (useNeuralNet)
{

datal = (datal + maxValueInputVect) * (2 * maxValueInput);
data2 = (data2 + maxValueInputVect) * (2 * maxValueInput);
data3 = (data3 + maxValueInputVect) * (2 * maxValueInput);

neuralNetX.evaluate([datal.x, data2.x, data3.x], out resultVect.x);
neuralNetY.evaluate([datal.y, data2.y, data3.y], out resultVect.y);
neuralNetZ.evaluate([datal.z, data2.z, data3.z], out resultVect.z);

resultVect = (resultVect * 2) - maxValueOutputVect;
} else {

    resultVectx = neuralNetX.evaluate(data1.x, data2.x, data3.x);
    resultVecty = neuralNetY.evaluate(data1.y, data2.y, data3.y);
    resultVectz = neuralNetZ.evaluate(data1.z, data2.z, data3.z);
}

transform.position = transform.position + (resultVect * moveSpeed);

[042] FIG. 5 Displays the flow chart of the use of head tracking technology that uses the portable device’s front facing camera to change position and rotation based on the location of the user’s face.

[043] When head tracking is used as a game controller, it can be straightforward what you can do to pan the virtual paddle in the game world in a code similar to this example:

[044]

    float scaleFactor = 1;

    paddleTransform.position = -HeadTracker.newPosition * scaleFactor;

    Vector3 localPosition = paddleTransform.localPosition;
    localPosition.y = -0.2f; //fixed value;
    paddleTransform.localPosition = localPosition;

[045] FIG. 6 Describes how head tracking technology can be used to train and calibrate a neural network that uses the linear acceleration data, since the head tracking processing has performance issues with current technology it can be used as reference for neural networks with valid data.
Vector3 speedVect = (Capturer.rawHeadTrackDataPos[endFrame].rawDataValue -
    Capturer.rawHeadTrackDataPos[initialFrame].rawDataValue) /
    (Capturer.rawHeadTrackDataPos[endFrame].timeframe -
    Capturer.rawHeadTrackDataPos[initialFrame].timeframe);

Vector3[] initialAcceleration = [Capturer.rawHeadTrackDataPos[initialFrame-1].rawDataValue,
    Capturer.rawHeadTrackDataPos[initialFrame].rawDataValue
    Capturer.rawHeadTrackDataPos[initialFrame+1].rawDataValue];

Vector3[] endAcceleration = [Capturer.rawHeadTrackDataPos[endFrame-1].rawDataValue,
    Capturer.rawHeadTrackDataPos[endFrame].rawDataValue
    Capturer.rawHeadTrackDataPos[endFrame+1].rawDataValue];

neuralNetX.Train([initialAcceleration[0].x, initialAcceleration[1].x, initialAcceleration[2].x],
    speedVect.x, 1000); //train begin movement

neuralNetX.Train([endAcceleration[0].x, endAcceleration[1].x, endAcceleration[2].x],
    0, 1000); //train end movement

neuralNetY.Train([initialAcceleration[0].y, initialAcceleration[1].y, initialAcceleration[2].y],
    speedVect.y, 1000); //train begin movement

neuralNetY.Train([endAcceleration[0].y, endAcceleration[1].y, endAcceleration[2].y],
    0, 1000); //train end movement

neuralNetZ.Train([initialAcceleration[0].z, initialAcceleration[1].z, initialAcceleration[2].z],
    speedVect.x, 1000); //train begin movement

neuralNetZ.Train([endAcceleration[0].z, endAcceleration[1].z, endAcceleration[2].z],
    0, 1000); //train end movement
FIG. 7 Shows another application of the computer programs to detect hit motions on game object using the game camera as a paddle, by detecting a quick motion in the raw captured data, either linear or rotational, and calculating a force value to be applied in a target game object in the virtual world.

For example, the "flick" movement along the X axis for pitch can be detected as follows:

```
Vector3 accelData1 = Capturer.rawGyroRotData[0].rawDataValue;
Vector3 accelData2 = Capturer.rawGyroRotData[1].rawDataValue;

if(accelData1.x > thresholdX && accelData2.x < thresholdX)
    PaddleControl.hitForce = force;
```

For a linear hit motion by the user in the Z axis, the code can be as follows:

```
float accelZ1 = Capturer.rawGyroAccelData[0].rawDataValue.z;
float accelZ2 = Capturer.rawGyroAccelData[1].rawDataValue.z;
float accelZ3 = Capturer.rawGyroAccelData[2].rawDataValue.z;

float diff1 = accelZ2 - accelZ1;
float diff2 = accelZ2 - accelZ3;

if(diff1 > thresholdZ && diff2 > thresholdZ)
    PaddleControl.hitForce = force;
```

This can also be applied with head tracking replacing the first three lines with:

```
float accelZ1 = Capturer.rawHeadTrackDataPos[0].rawDataValue.z;
```
float accelZ2 = Capturer.rawHeadTrackDataPos[1].rawDataValue.z;
float accelZ3 = Capturer.rawHeadTrackDataPos[2].rawDataValue.z;

[054] For the neural networks, the captured data must be normalized according to the
programming conventions of this algorithm before sending it to a neural network component
that is either trained or in the training phases and de-normalize the results to be fit to use in
controls of the game objects in the 3D world.

[055] The following code snippet illustrates the use when it's a linear hit using variable force. It
can also be applied with the head tracking movement data.

[056]
float accelZ1 = (Capturer.rawGyroAccelData[0].rawDataValue.z + maxValuelnput) * (2 *
maxValuelnput);
float accelZ2 = (Capturer.rawGyroAccelData[1].rawDataValue.z + maxValuelnput) * (2 *
maxValuelnput);
float accelZ3 = (Capturer.rawGyroAccelData[2].rawDataValue.z + maxValuelnput) * (2 *
maxValuelnput);

float result = 0;
network.evaluate([accelZ1, accelZ2, accelZ3], out result);
result = (result * 2) - maxValueOutput;
PaddleControl.hitForce = result * force;

[057] FIG. 8 802 illustrates a case of changes in rotation of 303 and will be reflected on 304 on
the corresponding rotation axis by reading data on the gyroscope inside the portable device.
While many programs outside this invention already can reflect the orientation of 303 in 304, this
invention uses the changes in orientation over a period of time to detect complete motions
performed by the user such as flips and spins and even more complex movements like rocking
or buoyancy. The algorithms of AI like fuzzy logic or neural networks as well as linear calculations are used for these mentioned motions as well as hit detection based on rotation speed and calculate a force to be applied in a virtual game object like a baseball. Wrist movements are used as base cases for the hit detections.

[058] FIG. 9 Shows the flow chart that the computer programs use to detect the rotational tricks performed on the real portable device as shown in FIG. 8.

[059] For the rotation gestures, a similar structure is required to store the data over time:

[060]
```csharp
public struct GyroRotationData{
    public Vector3 rotationData;
    public float timeframe;
}
```

[061] To provide storage of the data, a collection is required.

[062]
```csharp
public List<GyroRotationData> gyroRotationDataList = new List<GyroRotationData>();
```

[063] The linear approach takes a certain number of elements from the gyroRotationData list and evaluates them manually to determine if there has been a gesture. This is done by using predefined parameters that define the necessary conditions for the gesture. For example, to detect a 180 degrees rotation, the evaluation can be used as follows.

[064]
```csharp
float angles1 = Capturer.gyroRotationDataList[0].rotationData.x;
float angles2 = Capturer.gyroRotationDataList[1].rotationData.x;
float angles3 = Capturer.gyroRotationDataList[2].rotationData.x;
```
float diffl = angles2 - anglesl;
float diff2 = angles3 - angles2;

if (diffl > 88 && diff2 > 88) //use 88 degrees comparison instead of 90
    return true;
//an 180 degree move has been detected

[065] For fuzzy logic and neural network approaches, the same conventions are defined for hit
detection because each system uses them. Only the systems themselves are different since their
methods are different as well. The following sample code combines both methods separated by
an in game parameter that will define which method is used:

[066]
float anglesl = Capturer.gyroRotationDataList[0].rotationData.x;
float angles2 = Capturer.gyroRotationDataList[1].rotationData.x;
float angles3 = Capturer.gyroRotationDataList[2].rotationData.x;

float diffl = angles2 - anglesl;
float diff2 = angles3 - angles2;

float result = 0;
if (useNeuralNet)
    neuralNet.evaluate([diffl, diff2], out result);
else
    result = fuzzyLogic.evaluate(diffl, diff2);

if (result > 0.5) //half a radian
    //an 180 degree move has been detected

[067] Setting up several triggers within a time frame to be triggered in order allows for the
interpretation of complex movement by the user. An example of this would be a user twisting a
device back and forth like a princess’s wand. A gesture set to return true after a device is rotated
at least +50 degrees along the up and down axis and then -50 degrees and then +50 degrees
and then -50 degrees would then return a true value. If the user triggers a movement along the
other axes then this particular gesture recognition pattern would be invalidated. We define a
gesture to be composed of 1 or more thresholds which return a true value and a sequence in
which the thresholds are performed as well as a set of 0 or more triggers which could invalidate
the gesture.

[068]

```java
public class Gesture{
    List<gestureTrigger> gt;
    int[] lastGestureTrue = new int[3];
    float lastGestureTrueTime;
    int i;

    public bool CheckGesture(Vector3[] angles, float dataTime){
        for(i = 0; i < 3; i ++){ //for each axis
            gestureTrigger gtl = gt[lastGestureTrue[i]];
            //we have gone past the time threshold for the user to have moved from one
            //so reset it to start from the beginning of the gesture detection sequence of
            //we are within the time so check if it is a gesture
            if(gtl.isGesture(angles, i)){
                lastGestureTrue[i]++;
                //if>= then all triggers have returned true and this gesture has been
                recognized
                if(lastGestureTrue[i] >= gt.Count){
                    //its true, reset everything return true
                    Reset();
                }
            }
        }
        return false;
    }
}
```
return true;
}
public void Reset(){
    lastGestureTrue = new int[3] {0, 0, 0};
    lastGestureTrueTime = 0;
}
public class gestureTrigger{
    float[] threshHold; /*set the threshold for this trigger for example
    // = {88f, -30f, 45f};
    public float gestureTimeThreshold;
    public bool isGesture(Vector3[] angles, int axis){
        float diff1 = angles[l][axis] - angles[0][axis];
        float diff2 = angles[2][axis] - angles[l][axis];
        if(diff1 > threshA && diff2 > threshHold[ax])
            return true;
        return false;
    }
}

[069] When 1:1 rotation is not desired, it is useful to scale the rotation rate of the gyrometers
or accelerometers to achieve desired results. For example, to rotate the in-game camera 90 degrees by just moving the actual device 45 degrees, just duplicating the rotation rate is needed.

\[ \text{cameraTransform.localEulerAngles.z} += \text{Time.deltaTime} \times \text{Capturer.rawGyroAccelData} \times \text{OJ.rawDataValue.z} \times \text{scale}; \]

[071] FIG. 10 depicts a solution when orientations of the portable device can also be used to set both position and rotation of the virtual game object. 1002 is the classic representation of pivot at the center of the portable device as well as the game object that is changed by the detection algorithms where only the rotation of the virtual game object is changed. However, by creating an empty game object that will perform as the base parent object 1003 and another empty game object that will act as a pivot or center of rotations 1004 in a position away from the center of the device we can change the rotation and position of the game object or game camera 1005 that is not a child of the pivot object 1004 and can be at the same position as 1003 or a separate position for different motion effects that depend more on the game or application design. Also, the pivot object can be at any rotation to add different effects like reverse rotations, or rotate in different axes thanks to the freedom of this simple design.

[072] FIG. 11 illustrates how the local motion coordinates change depending on the orientation of the portable device and it can be detected and used with different methods. By using simple game object hierarchy the movement and rotation axes 1102 change depending of the orientation of the device. This removes the requirement that the user has to be in a specific pose in order to control the virtual objects or the game camera with the portable device.
FIG. 12 illustrates a solution for the different ways a user can use the game or application and perform linear or rotational motions to control the game objects such as a game camera 1203. This is used by having a parent object of the game camera 1202 where the rotation of the device will be transformed to local orientation of the camera. By using an external signal to call the reorientation, the rotation data of 1203 is stored temporarily in an initial state 1205 to then invert and applied to 1202 so that 1203 resets to an initial orientation that the user needs to keep using the game or application in a new state 1206. This allows users to be in any pose as they desire like standing up or laying down and keep playing without interruptions.

The code to re-center a game camera is as follows:

```java
private Transform cameraTransform; // The transform of the game camera,
private Transform camParent; // The transform of the camera parent.

public void Recenter()
{
    camParent.localRotation = Quaternion.Inverse(cameraTransform.localRotation);
}
```

FIG. 13 Describes a multiplayer session between users where each user sees the game from their perspective and see the opponent on the opposite side like in a table tennis match. In order to accomplish this a center point will act as a mirror reference point 1302 that will be an exact global reference point but rotated 180 degrees. So when a second user connects to the game, his game object 1304 will be the child of 1302 and will control in the exact same way like the first user 1303. The users can be in the same network or in different parts of the world. The
center and mirror points will act as virtual references and the only synchronization does not need any real world reference and the re-centering programs will be transparent to each as explained in FIG. 12. To the other player it will appear as they are just in front of them when they can be in any pose in the real world in any network.

[077] The code that allows this accomplishment goes as follows:

[078]

```csharp
private Transform cameraTransform; //The transform of the game camera,
private Transform localAuxiliaryTrs; //The transform of the camera's auxiliary obj.
private Transform otherTransform; //The auxiliary transform of the opponent's device.
...

void Update()
{
    ...
    localAuxiliaryTrs.rotation = cameraTransform.rotation;
    SendToNetwork("GetOtherPlayerRotation", cameraTransform.localRotation);
}
...

public void GetOtherPlayerRotation(Quaternion localRotation)
{
    otherTransform.localRotation = localRotation;
}
```

[079] With the above embodiments in mind. It should be also noted that this invention can also be applied to any device that contains accelerometers and gyroscopes such as dedicated portable devices and peripherals that can interact with such devices. The inventions also work when these portable devices can be used as remote controllers for other devices when the programming allows such behavior.
WE CLAIM:

1. A method of detecting complex linear and rotational movements based on the data obtained from the gyroscope and accelerometers using both linear and artificial intelligence computer programs based on sampling data for a period of time and operate on that data to perform any desired action in an application or game such actions could be for either movement or hit detection for any object in the virtual space, the method includes:

   data sampling of the device's accelerometer and gyroscope data with variable sampling rate and time frame to be stored in a defined set for other programs to use; use;

   using linear calculations to detect peak or valley values of linear motions to detect start and end of movements in any axis without the need to detect a device's position in the real world space;

   using fuzzy logic artificial intelligence methods to detect movements and hit motions based on a set of rules that vary depending on the type and scope of the application or game;

   using the data sample as a whole and filters noise data that affects the accuracy of the detection that the linear method considers relevant when in reality it is not;

   using neural network algorithms that operate on the sampling data and perform a similar detection of motion and hit as the fuzzy logic algorithm with the difference being that the neural network will be trained based on multiple user configuration and testing in both consumer and quality control sides;

   storing the learning parameters in a cloud environment and constantly modifying the learning parameters based on user feedback and application performance;

   storing a set of parameters for the neural networks in accordance with age group and...
application needs to avoid overlap and optimize behavior for each application or game;

and

using head tracking algorithms on portable devices that contain a front facing camera to not only move virtual objects in the 3D space but to use the changes in position over time to calculate forces and speed to be applied to other game objects like in hit detection and complex movement calculations.

2. The method of using the motions described in claim 1 to detect hits and for catching and throwing actions between users or in single use mode.

3. The method of Claim 1 used to detect trick moves such as flips, jumps, and spins.

4. A configurable scale ratio and order of real world movements to be reflected on virtual world objects that depend on the scope and design of the application or game by using simple motions performed on the portable device such as:

   a windshield wiper motion where the game object rotates around a center or pivot object so that it not only changes orientation but also moves in the 3D space and in the real world with just a rotation of the device; and

   reverse or different axis motions by changing which axis of the real world is influenced by a determined axis in the real world to achieve different effects like confusion status in a role playing game or a simpler control of the game object for a specific application.
5. A method that allows the re-centering of a game object or camera to an initial state in the virtual world when the user changes pose in the real world to keep playing by using parent objects that invert the rotation that represents the orientation of the portable device, thus allowing the game object or camera to return to the original state without pausing or restarting the application or game.

6. A method that allows a multiplayer session while keeping the controls of the single player mode by creating virtual reference points that are located in the exact position as the reference point in single player mode but with a different rotation so it can become a reference point for other users. This is achieved by using these virtual reference points as parent objects of the other users and all changes in positions and rotations the user performs will be reflected in the virtual world using the parent object as reference. This saves time and development costs by keeping the same code using local movement and rotation calculations instead of global.

7. The method described in claim 6 also has the benefit of transmitting local position and rotation that other players see naturally while not affecting the changes of the game world in the multiplayer session.
start

Obtain Head Tracking Data

Calculate New Position and Rotation Of Game Object Based on Tracking Data

Smooth Changes with Interpolation

end

FIG. 5
start

Obtain Raw Linear or Rotational Data

Check for Hit Motion (Fuzzy Logic, Neural Network, Linear or Head Tracking)

Hit Detected?

Yes

Calculate Force Scalar

Apply Force on Direction Vector to Ball

end

FIG. 7
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
G06F 3/01(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 3/01; G09G 5/08; G08B 1/08; A63F 9/24; A63F 13/00; G09G 5/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic database consulted during the international search (name of database and, where practicable, search terms used)
eKOMPASS/KIPO internal & Keywords: movement, virtual space, fuzzy logic algorithms, neural network algorithms, linear algorithms, head tracking algorithms, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>WO 2013-166251 AI (AI GOLF, LLC) 07 November 2013 See paragraphs [0040], [0048], [0070], [0074], [0090], [0150] and figures 1-7(b).</td>
<td>1-3</td>
</tr>
<tr>
<td>Y</td>
<td>US 2009-0219224 AI (JOHANNES ELG) 03 September 2009 See paragraphs [0037], [0041]-[0045] and figure 1.</td>
<td>1-3</td>
</tr>
<tr>
<td>A</td>
<td>US 2012-0299827 AI (DAN OSBORN et al.) 29 November 2012 See paragraphs [0012]-[0034] and figures 1-4.</td>
<td>1-3</td>
</tr>
<tr>
<td>A</td>
<td>US 2009-0209343 AI (ERIC FOXLIN et al.) 20 August 2009 See paragraphs [0037]-[0074] and figures 1-2.</td>
<td>1-3</td>
</tr>
<tr>
<td>A</td>
<td>US 2012-0105225 AI (TWO VALTONEN) 03 May 2012 See paragraphs [0075]-[0085] and figures 3-4.</td>
<td>1-3</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
05 October 2015 (05. 10.2015)

Date of mailing of the international search report
06 October 2015 (06.10.2015)

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsan-ro, Seo-gu, Daejeon Metropolitan City, 35208, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
BYUN, Sung Cheal
Telephone No. +82-42-481-8262

Form PCT/ISA/210 (second sheet) (January 2015)
This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claims Nos.: 
   because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claims Nos.: 
   because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claims Nos.: 
   because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

This International Searching Authority found multiple inventions in this international application, as follows:

I. Claims 1-3 are directed to a method of detecting complex linear and rotational movements based on the data obtained from the gyroscope and accelerometers using both linear and artificial intelligence computer programs based on sampling data for a period of time.

II. Claim 4 is directed to a configurable scale ratio and order of real world movements to be reflected on virtual world objects that depend on the scope and design of the application or game by using simple motions performed on the portable device.

III. Claim 5 is directed to a method that allows the re-centering of a game object or camera to an initial state in the virtual world.

IV. Claims 6-7 are directed to a method that allows a multiplayer session while keeping the controls of the single player mode by creating virtual reference points.

1. ☐ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.

2. ☐ As all searchable claims could be searched without effort justifying an additional fees, this Authority did not invite payment of any additional fees.

3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.: 1-3

Remark on Protest

☐ The additional search fees were accompanied by the applicant’s protest and, where applicable, the payment of a protest fee.

☐ The additional search fees were accompanied by the applicant’s protest but the applicable protest fee was not paid within the time limit specified in the invitation.

☐ No protest accompanied the payment of additional search fees.
<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AU 2013-256266 B2</td>
<td>09/04/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CA 2869785 Al</td>
<td>07/11/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN 104394949 A</td>
<td>04/03/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EP 2844359 Al</td>
<td>11/03/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2015-525085 A</td>
<td>03/09/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KR 10-2015-0006463 A</td>
<td>16/01/2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 2013-0296048 Al</td>
<td>07/11/2013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>US 9022870 B2</td>
<td>05/05/2015</td>
</tr>
<tr>
<td>US 2012-0105225 Al</td>
<td>03/05/2012</td>
<td>US 8456298 B2</td>
<td>04/06/2013</td>
</tr>
</tbody>
</table>