УДК 004.42

РАЗРАБОТКА ГИРОСКОП КОНТРОЛЛЕРА ДЛЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА ПЛАТФОРМЕ UNITY

© 2018 О. Ю. Лазарева, П. С. Афанасьев

Московский политехнический университет
Высшая школа печати и медиаиндустрии (г. Москва, Россия)

В работе рассматриваются проблемы, возникающие при разработке гироскоп контроллера для мобильного приложения на платформе Unity, и пути их решения. Выделены преимущества использования гироскоп контроллера в мобильных приложениях.

Ключевые слова: гироскоп, гироскоп контроллер, мобильное приложение, игровое приложение, движок Unity

Современные мобильные устройства оснащаются большим количеством интересных функций и модулей. Одним из них является гироскоп, и если совсем недавно это было новинкой, то сегодня он используется повсеместно и удивить наличием этого модуля современного пользователя сложно. Гироскоп – это устройство, которое реагирует на изменения углов ориентации тела, на которое оно устанавливается. Гиродатчик используется во многих серьезных сферах науки и техники: судоходстве, космонавтике, авиации, в производстве техники бытового назначения, игрушек, и, конечно же, мобильных телефонов.

Гироскоп не следует путать с акселерометром. Акселерометр – это датчик, который определяет угол наклона электронного устройства по отношению к земной поверхности. Программное обеспечение устройства получает данные об угле наклона от акселерометра и поворачивает изображение на дисплее, т. е. назначение акселерометра заключается в отслеживании поворотов устройства в пространстве. А гироскоп же фиксирует скорость перемещения телефона в пространстве и определяет стороны света. Функциональность у этих двух устройств похожа, а гиродатчик можно даже назвать улучшенным акселерометром.

Применение гиродатчика в смартфонах открыло абсолютно новые возможности. Пользователь может по достоинству оценить функциональность этого устройства в своем мобильном телефоне. Например, отвечать на звонки, просматривать фото, изображения, переворачивать страницы в электронной книге, переключать музыку в плеере можно с помощью элементарного встряхивания телефона. Невероятно удобен гиродатчик и в калькуляторе: вертикальное положение телефона позволяет выполнять минимальное количество функций: складывать, вычитать, умножать и делить. Но, если пользователь повернет телефон на 90 градусов, то калькулятор перейдет в расширенный режим: на дисплее смартфона появится масса дополнительных математических операций и функций.

В некоторых операционных системах встряхивание телефона может запустить обновление Bluetooth. Гиродатчик используется в работе специфических программ, предназначающихся для измерения углов наклона и уровня. Также очень удобен гироскоп, когда необходимо определить местоположение пользователя на незнакомой местности. При помощи GPS-навигации можно использовать карту, которая при любом повороте человека меняет свое направление в ту же сторону. Если пользователь повернётся лицом в другую сторону, это автоматически отобразится на карте. Такой помощник значительно упрощает ориентирование на местности и станет чрезвычайно полезным для людей, увлечённых активными видами отдыха.

Помимо преимуществ, описанных выше, существуют и недостатки, из-за которых некоторые пользователи предпочитают отказаться от использования гироскопа в своем смартфоне и отключают его. Недостатком можно, например, считать, что некоторые приложения реагируют на изменения положения в пространстве с небольшим

Лазарева Ольга Юрьевна – Московский политехнический университет Высшая школа печати и медиаиндустрии, доцент кафедры информатики и информационных технологий, к. т. н., lazarevaoy@gmail.com.

Афанасьев Павел Сергеевич – Московский политехнический университет Высшая школа печати и медиаиндустрии, студент Института принтмедиа и информационных технологий, pavel.97@list.ru.

опозданием. Если пользователь, читая лёжа электронную книгу, при этом будет менять свою позу, то гироскоп-датчик и приложение, связанное с устройством, также будут менять ориентацию страницы. Это доставляет определенные неудобства.

Конечно же, чаще всего, гироскоп используют любители игр. Наличие гироскопа переводит процесс игры в другое качество. С ним можно управлять не только поворотами, но и скоростью поворотов. Любое движение героя на дисплее становится более точным, реалистичным. Этот датчик совершенно необходим для многих жанров игр: гонок, шутеров и т. д. Именно он помогает навести прицел, повернуть руль автомобиля и т. д.

Для упрощения разработки игровых приложений часто используются различные игровые движки. Одним из распространённых игровых движков является Unity. Это кроссплатформенная среда разработки игр, позволяющая создавать приложения, работающие под многими операционными системами, включающими персональные компьютеры, игровые консоли, мобильные устройства, Интернет-приложения и другие. Основными преимуществами Unity являются наличие визуальной среды разработки, межплатформенной поддержки и модульной системы компонентов. К недостаткам относят появление сложностей при работе с многокомпонентными схемами и затруднения при подключении внешних библиотек. Движок поддерживает два скриптовых языка: C# и модификацию JavaScript.

Далее рассматривается процесс разработки гироскоп контроллера для мобильного приложения на языке C# с использованием игрового движка Unity.

Гироскоп дает широкий спектр интересных вариаций управления, но когда доходит дело до его интеграции и реализации контроллера камеры, возникают трудности.

Должны выполняться определённые правила:

1. Экран устройства должен работать как «окно» в виртуальный мир. Поворачивая мобильное устройство, пользователь должен осматривать этот мир.

2. Контроллер должен поддерживать функцию автоповорота и поддерживать все ориентации устройства.

3. Верхняя ось виртуального мира должна соответствовать верхней оси в реальном мире и должна быть осуществлена рекалибровка горизонтального вращения.

Чтобы задать свойства вращений, используется класс Quaternion. Один из методов этого класса Euler() позволяет задать вращение в градусах по трем осям (X, Y, Z).

Для начала вводятся множественные необходимые характеристики: начальная ориентация устройства baseOrientation, калибровка calibration и т. д.

Код инициализации характеристик представлен ниже:

```
private const float lowPassFilterFactor = 0.2f;
private readonly Quaternion baseIdentity = Quaternion.Euler(90, 0, 0);
private readonly Quaternion landscapeRight = Quaternion.Euler(0, 0, 90);
private readonly Quaternion landscapeLeft = Quaternion.Euler(0, 0, -90);
private readonly Quaternion upsideDown = Quaternion.Euler(0, 0, 180);
private Quaternion cameraBase = Quaternion.identity;
private Quaternion calibration = Quaternion.identity;
private Quaternion baseOrientation = Quaternion.Euler(90, 0, 0);
private Quaternion baseOrientationRotationFix = Quaternion.identity;
private Quaternion referanceRotation = Quaternion.identity;
```

Включить гироскоп можно с помощью следующей строчки:

```
Input.gyro.enabled = true;
```

Но при этом возникнут несколько проблем, мешающие правильному функционированию гироскопа.

Первая проблема, с которой можно столкнуться, это разный тип систем координат, использующийся в мобильных устройствах и Unity: левосторонняя и правосторонняя. Для того, чтобы преобразовать кватернионы из одной системы координат в другую, используется функция ConvertRotation(). Код функции представлен ниже:

```
private static Quaternion ConvertRotation(Quaternion q)
{
return new Quaternion(q.x, q.y, -q.z, -q.w);
}
```

Следующая проблема – поддержка различных ориентаций устройства, но разработчиками она была решена в 4 версии Unity. Поэтому в функции GetRotFix() воз-

вращается тоже самое значение ориентации устройства. Код функции представлен ниже:

```
private Quaternion GetRotFix()
{
return Quaternion.identity;
}
```

Также необходимо, чтобы во время работы гироскопа менялась калибровка, базовое вращение камеры, базовая ориентация устройства и пересчитывался поворот камеры.

Для смены базовой ориентации устройства, используется метод ResetBaseOrientation(). Чтобы получить новое значение ориентации, нужно умножить предыдущее его значение baseIdentity на значение поворота baseOrientationRotationFix. Код реализации этого метода представлен ниже:

```
private void ResetBaseOrientation()
{
baseOrientationRotationFix = GetRotFix();
baseOrientation = baseOrientationRotationFix * baseIdentity;
}
```

Для изменения калибровки используется метод UpdateCalibration. В переменную fw записывается значение, равное произведению положения в пространстве камеры на отрицательный вектор с координатами (0,0,1). Если координата по оси Z не равна 0, то калибровка должна поменяться, иначе не изменяется. Код реализации этого метода представлен ниже:

```
private void UpdateCalibration()
{
var fw = (Input.gyro.attitude) *
(-Vector3.forward);
fw.z = 0;
if (fw == Vector3.zero)
{
calibration = Quaternion.identity;
}
else
{
calibration = (Quaternion.FromToRotation
(baseOrientationRotationFix * Vector3.up, fw));
}
}
```

Для изменения базового вращения камеры используется метод UpdateCameraBaseRotation(). В переменную fw записывается значение трансформации по оси Y. Если оно равно 0, то вращение камеры не происходит, иначе происходит на основе положения камеры в пространстве и значения трансформации как параметров метода FromToRotation() класса Quaternion. Код реализации этого метода представлен ниже:

```
private void UpdateCameraBaseRotation()
{
var fw = transform.forward;
fw.y = 0;
if (fw == Vector3.zero)
{
cameraBase = Quaternion.identity;
}
else
{
cameraBase = Quaternion.FromToRotation(Vector3.forward, fw);
}
}
```

Код функции RecalculateReferenceRotation(), отвечающей за пересчёт поворота, представлен ниже:

```
private void RecalculateReferenceRotation()
{
referanceRotation = Quaternion.Inverse(baseOrientation) * Quaternion.Inverse(calibration);
}
```

Наконец, рассчитывается общее изменение положения камеры в каждый момент времени. Для этого используется метод Slerp() класса Quaternion. Этот метод позволяет производить сферическую интерполяцию между параметрами A и B. В данном случае, параметром A является положение камеры в пространстве в прошедший момент времени, а параметром B – произведение значений характеристик, полученных ранее в других методах. Третий параметр, которым является характеристика lowPassFilterFactor, показывает, на сколько резко должно произойти изменение положения камеры в пространстве.

Код служебной функции Update(), отвечающей за изменение положения камеры в пространстве, представлен ниже:

```
protected void Update()
{transform.rotation = Quaternion.Slerp(transform.rotation,
cameraBase * (ConvertRotation(referanceRotation * Input.gyro.attitude) * GetRotFix()), lowPassFilterFactor);
}
```

Приведенный код позволяет решить проблемы, возникающие при разработке ги-

роскоп контроллера для мобильного приложения на платформе Unity. Пользователь, используя своё мобильное устройство, может насладиться виртуальной (игровой) реальностью. Таким же образом, но уже не в игре, можно использовать гироскоп для формирования дополненной реальности.

ЛИТЕРАТУРА

1. Как написать гироскоп контроллер на Unity // Создание игр для начинающих – Разработка игр Unity. URL: https://gcup.ru/publ/gamedev/kak_napisat_giroskop_kontroller_na_unity/1-1-0-494 (дата обращения: 01.03.2018).

2. Что такое гироскоп в телефоне? // Официальный сайт российского производителя смартфонов Хайскрин. URL: http://highscreen.ru/news/giroskop (дата обращения: 01.03.2018).

3. Gyroscope controller that works with any device orientation. // GitHub. URL: https://github.com/kibotu/net.kibotu.sandbox.unity.dragnslay/blob/master/unity/Assets/Scripts/GyroController.cs (дата обращения: 01.03.2018).

4. Unity // Unity Official site. URL: https://unity3d.com/ru (дата обращения: 01.03.2018).

DEVELOPMENT OF THE GYROSCOPE CONTROLLER FOR MOBILE APPLICATIONS ON THE UNITY PLATFORM

© 2018 O. Yu. Lazareva, P. S. Afanasyev

Moscow Polytechnic University
Higher School of Print and Media Industry (Moscow, Russia)

The paper discusses the problems that arise when developing a gyroscope controller for a mobile application on the Unity platform, and ways to solve them. The advantages of using a gyroscope controller in mobile applications are highlighted.

Key words: gyroscope, gyroscope controller, mobile application, gaming application, Unity engine.