DOI 10.37882/2223-2966.2023.06.23

ИСПОЛЬЗОВАНИЕ ПРОСТЕЙШИХ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ И ГЕОМЕТРИЧЕСКИХ ЗАВИСИМОСТЕЙ В РАЗРАБОТКЕ ВИДЕОИГР

USING SIMPLE MATHEMATICAL FUNCTIONS AND GEOMETRIC DEPENDENCES IN VIDEO GAME DEVELOPMENT

K. Kuzminsky
T. Ryasyanen
T. Ulengova

Summary: This article discusses the areas and examples of the application of mathematical functions and geometry in the development of video games. On the basis of the study, the main theses, advantages and disadvantages of using mathematics in the creation of games are formulated, examples of the use of mathematical functions and geometry in the development of video games are presented and analyzed.

Keywords: development, game application, mathematics, mathematical function, geometry.

Кузминский Кирилл Витальевич
Тихоокеанский Государственный Университет
2017100693@pnu.edu.ru
Ряйсянен Татьяна Николаевна
старший преподаватель,
Тихоокеанский Государственный Университет
000512@pnu.edu.ru
Уленгова Татьяна Георгиевна
старший преподаватель,
Тихоокеанский Государственный Университет
000516@pnu.edu.ru

Аннотация. В данной статье рассмотрены области и примеры применения математических функций и геометрии в разработке видеоигр. На основании проведенного исследования сформулированы основные тезисы, преимущества и недостатки использования математики при создании игр, изложены и проанализированы примеры применения математических функций и геометрии в разработке видеоигр.

Ключевые слова: разработка, игровое приложение, математика, математическая функция, геометрия.

В каждой игре разработчик активно оперирует различным функционалом, вложенным в игровой движок (в наших примерах движок Unity), для непосредственного создания игрового процесса. Но в некоторых случаях, например, при ограниченных временных ресурсах или за недостатком опыта, разработчику гораздо проще воспользоваться математикой для представления игрового процесса в виде простейших функций, чтобы в дальнейшем переложить это в понятный программный код. В рамках проекта «Ready, Steady, Swipe!» мы столкнулись с аналогичной ситуацией, когда разработка геймплея на основе геометрических вычислений оказалась подходящей как со стороны затраченного времени, так и со стороны загрузки вычислительных мощностей мобильного телефона.

Игра «Ready, Steady, Swipe!» представляет собой трехмерную казуальную аркаду. В данной игре игрок управляет кубиком, некоторые элементы которого подсвечены определенным цветом. Задача игрока — перемещать кубик на соседние клетки соответствующих ему цветов.

На этапе разработки концептуальной документации было принято решение о формате передвижения игрового кубика на соседние клетки игрового поля — модель должна была плавно и равномерно перевалиться на другую грань, опираясь при этом на нижнее ребро модели.

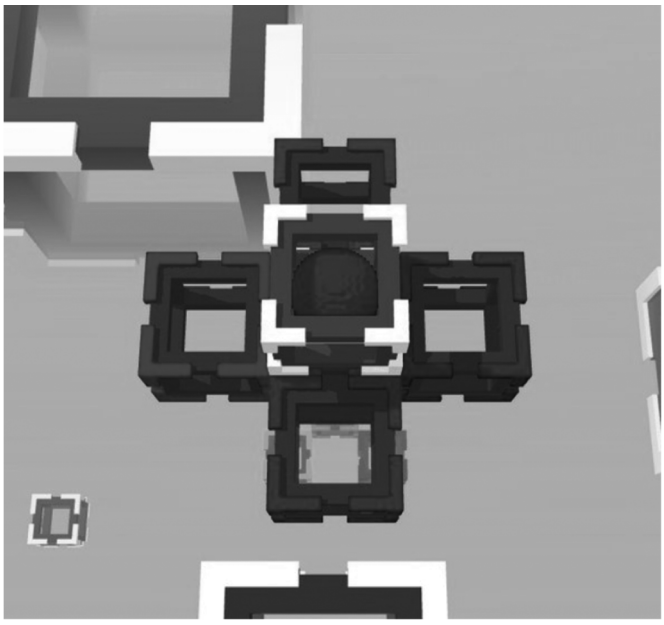

Рис. 1. Игровое поле «Ready, Steady, Swipe!»

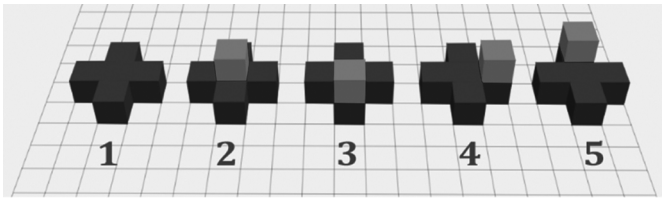

Рис. 2. Черновой концепт геймплея «Ready, Steady, Swipe!»

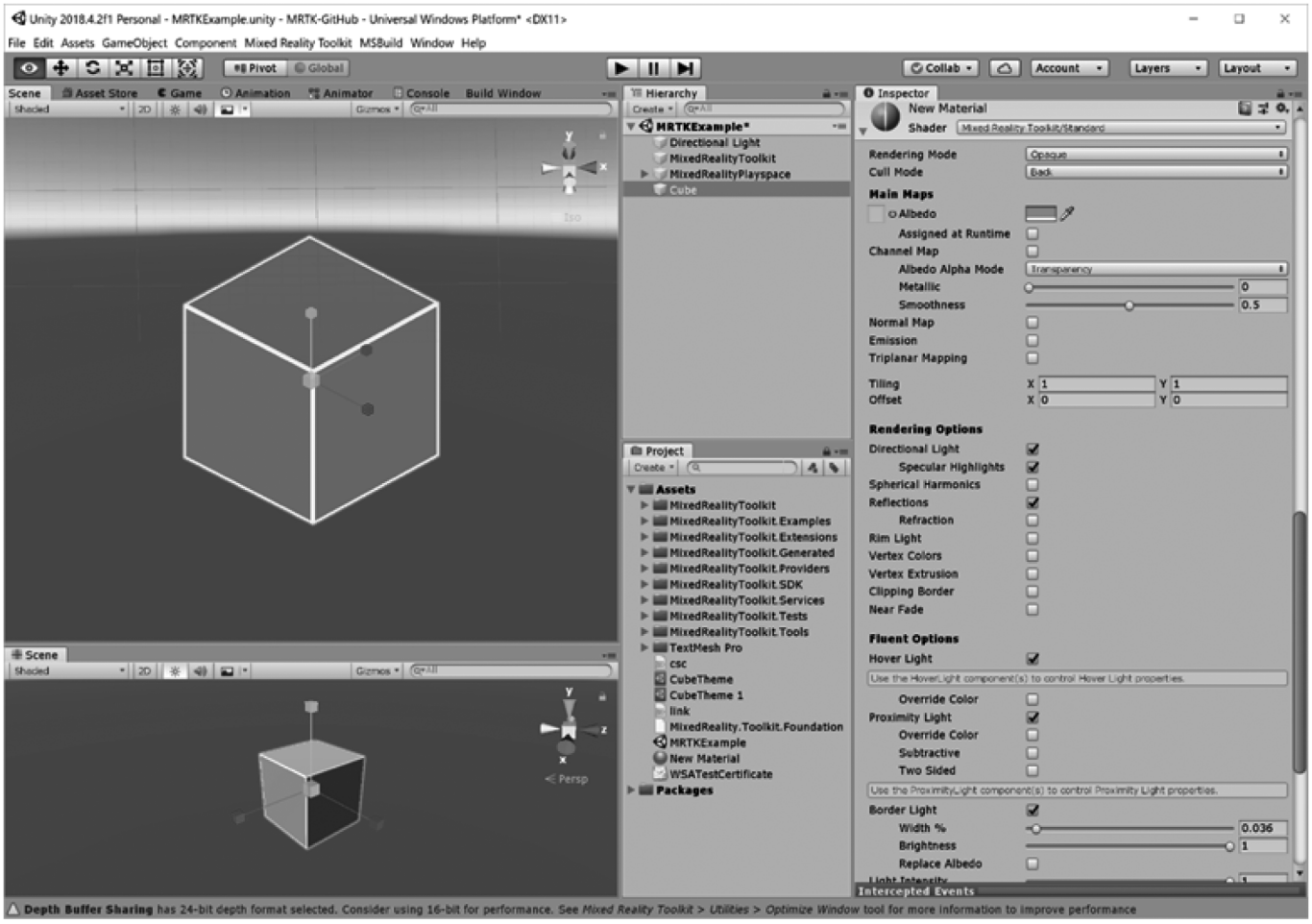

Рис. 3. Прототип модели в среде разработки Unity

На этапе разработки прототипа игры первоначально было принято решение воспользоваться внутренними возможностями движка Unity и выполнять вращение модели, основываясь на векторах и изменении параметров кватернионов игровой модели.

Кватернион представляет собой вращение на определенный угол вокруг произвольной оси. Любое вращение в Эйлеровом пространстве можно задать с помощью кватерниона.

В Unity вращения также задаются в кватернионах (Quaternion) не смотря на то, что есть доступ к Эйлеровым углам. Эйлеровы углы вычисляются из кватерниона при чтении и из Эйлеровых углов создается кватернион при присвоении.

Самое большое преимущество кватернионов — интерполяция. Кватернионы могут интерполироваться с помощью сферической линейной интерполяции (SLERP). Данный вид интерполяции позволяет найти кратчайший поворот на поверхности сферы.

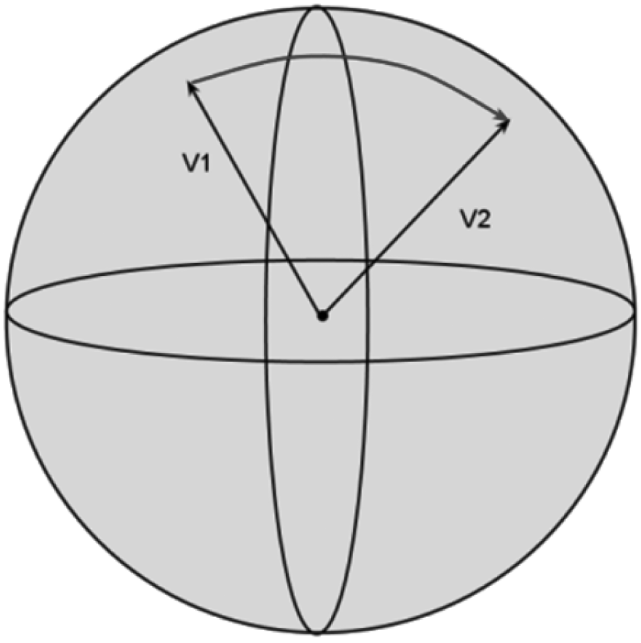

Рис. 4. Сферическая линейная интерполяция

После нескольких неудачных попыток интеграции данного метода в наш концепт мы заметили, что созда-

ние анимации подобным способом негативно влияет на производительность приложения, а также требует углубленного изучения применения кватернионов на основе игрового движка Unity при острой нехватке подходящей пояснительной документации. Тогда нами был выдвинуты следующие тезисы, которые в дальнейшем перешли в условия и ограничения для поиска решений подобных задач:

1. Метод создания анимации в масштабах конкретного проекта должен быть оптимальным с точки зрения производительности, по сравнению с аналогами;
2. Метод анимирования должен быть простым и интуитивно понятным для полноценного и качественного интегрирования в программный код приложения;
3. Метод анимирования должен быть точным и не допускать высоких погрешностей при вычислении позиций модели для минимизации появления ошибок как в программной, так и визуальной части приложения;

На основании данных условий было принято решение о поиске математической интерпретации требуемой анимации, который не требовал бы много ресурсов и усложнения программного кода или подхода к конкретной реализации приложения. В дальнейшем полученный метод был совмещен с векторным методом, что дало оптимальный вариант решения, удовлетворяющий условиям проекта.

В качестве базиса была рассмотрена модель вращения квадрата по касательной окружности относительно центра квадрата. В этом случае нижний угол квадрата (она же нижнее ребро трехмерной модели) является центром вращения, а остальные вершины квадрата перемещались в условиях жесткого сцепления с центром вращения и геометрическим центром самого квадрата.

Расчет модели перемещения квадрата производился на основе двух положений: исходного и срединного, когда геометрический центр квадрата находится ровно вертикально над центром описанной им окружности. Далее были введены номинальные размеры, где:

a — длина ребра квадрата,

R — радиус окружности, описанной геометрическим центром квадрата при вращении.

Геометрический центр квадрата был целенаправленно взят определяющей точкой, так как именно относительно него в дальнейшем будет проще построить вращение трехмерной модели в одной плоскости, учитывая перемещения всех вершин игрового куба.

После построения геометрической схемы предполагаемой анимации мы произвели вывод формулы, в которой значение координаты Y в двумерной системе прямо зависит от текущего значения координаты X. Координата X, в свою очередь, зависит от желаемого времени изменения положения кубика, что позволяет легко регулировать скорость течения анимации. Итогом мы получили систему вида

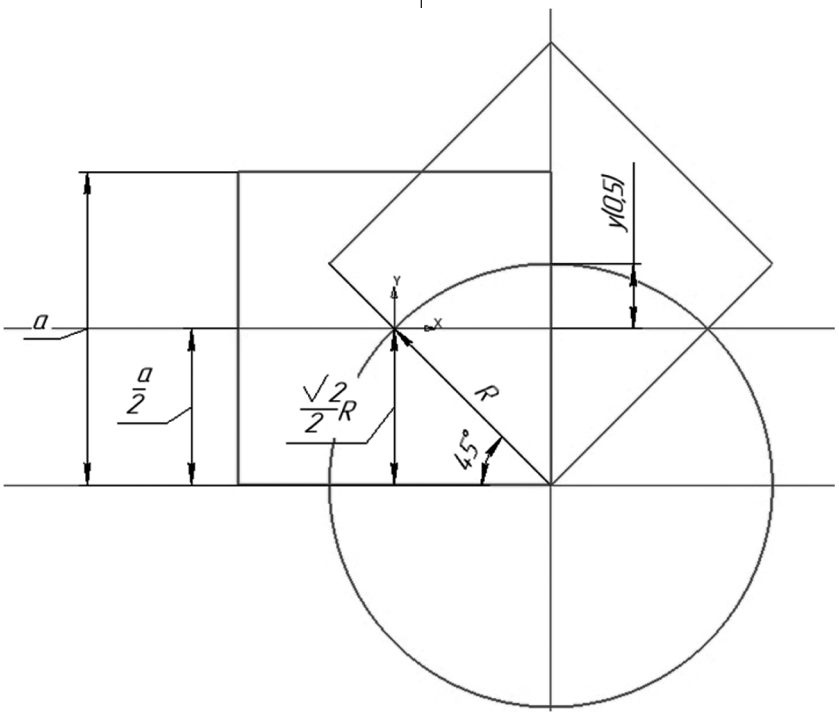

Рис. 5. Геометрия перемещения игровой модели

```
void CubikMove(){

    if (Naprav == 1) {

        //rigth

        x = Mathf.Lerp (0, 1, t);

        y = Mathf.Sqrt (0.5f - ((x - 0.5f) * (x - 0.5f))) - 0.5f;

        transform.position = new Vector3 (GhostX.x + x, y, transform.position.z);
```

Рис. 6. Полученная формула в программной интерпретации

$$\begin{cases} x = x(t), \text{ где } x = 0..1 \\ y = \sqrt{0.5 - (x - 0.5)^2} - 0.5 \end{cases}$$

Полученное выражение было перенесено в программный код и в дальнейшем протестировано на прототипе, где формула продемонстрировала желаемый результат.

Полученный результат оказался искомым и оптимальным как с программной точки зрения, так и с точки зрения визуального отображения перемещения внутри игровой модели.

Опыт в использовании простейших математическим формул мы использовали и в другом нашем проекте, «Sinusoid!». Игра представляет собой двумерную казуальную аркаду, в которой нужно управлять точкой. Задача игрока — уворачиваться от враждебных объектов.

Расчет перемещения точки происходит по функции синуса, от одной стороны экрана к другой по оси *X*. При нажатии на экран точка начинает движения по оси *Y*.

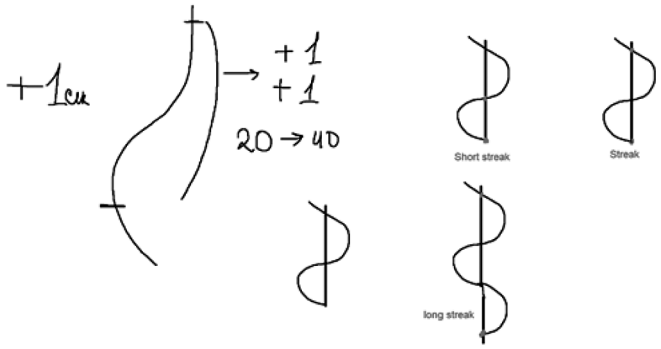

Рис. 7. Прототип механики Sinusoid!

Благодаря этой простейшей функции математики получился весьма затягивающий геймплей, который позволяет возвращаться игроку снова и снова.

Работа выполнена при поддержке Хабаровского отделения регионального научно-образовательного центра «Дальневосточный центр математических исследований» (допсоглашение с Минобрнауки от 16 февраля 2023 г. № 075-02-2023-932)

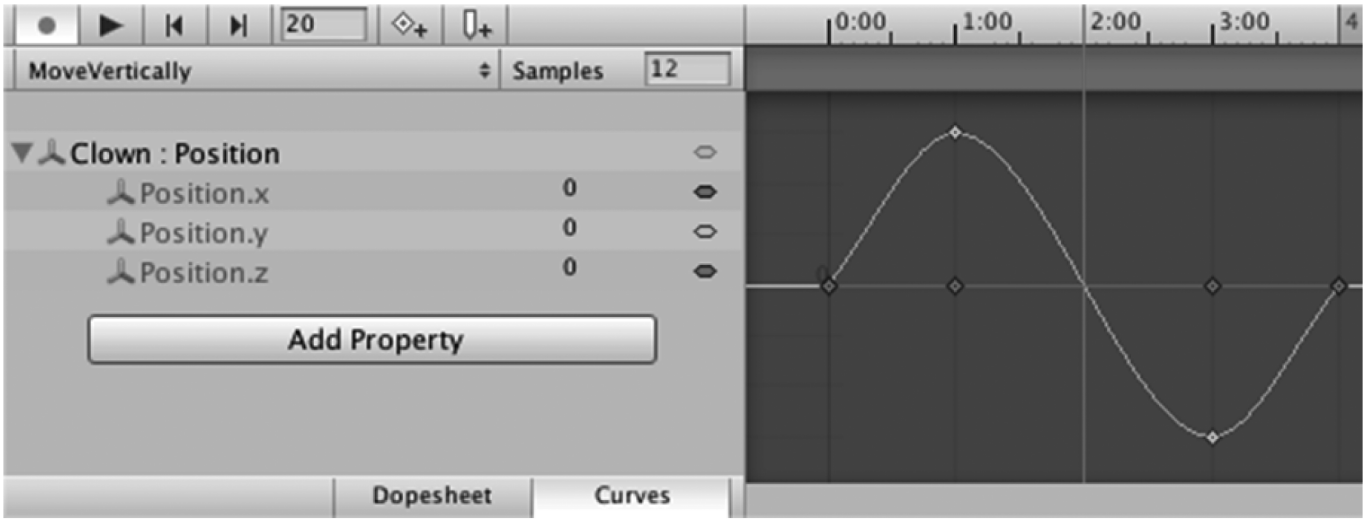

Рис. 8. Прототип функции в среде разработки Unity

Рис. 9. Изображение геймплея Sinusiod!

Вывод

Таким образом, на собственном опыте была изучена возможность использования простейших математических функция и геометрических зависимостей в разработке видеоигр. Благодаря использованию математических функций мы оптимизировали проект, для более быстрого отклика приложения для мобильных устройств, упростили вид программного кода для упрощения будущих изменений в приложении и сделали достаточно интересный геймплей.

ЛИТЕРАТУРА

1. Математика в Gamedev по-простому. Векторы и интегралы [Электронный ресурс] Режим доступа: [https://habr.com/ru/post/430146/]
2. Официальный сайт UNITY [Электронный ресурс] Режим доступа: [https://unity.com/ru]

© Кузминский Кирилл Витальевич (2017100693@pnu.edu.ru), Ряйсянен Татьяна Николаевна (000512@pnu.edu.ru);
Уленгова Татьяна Георгиевна (000516@pnu.edu.ru).
Журнал «Современная наука: актуальные проблемы теории и практики»