Математика в компьютерных играх

# **Содержание**

Введение 3

Глава 1 – Начало 4

Глава 2 – Треугольник между игроками 6

Глава 3– Расстояние между игроками 7

Глава 4 – Угол при повороте 8

Глава 5 – Движение по прямой 11

Глава 5.1 - Нахождение градиента 11

Глава 5.2 - Движение игрока 12

Глава 6 - Пример из настоящей видеоигры 13

Глава 7 – Прыжок по параболе 14

Глава 7.1 - Вычисление параболы 14

Глава 7.2 - Пример использования параболы 15

Заключение 16

# **Введение**

Прежде всего, что такое компьютерные игры? Для многих это просто красочные фигуры, которые двигаются по экрану. Однако, на самом деле это код, который состоит из бесконечных строк и способен управлять единицами и нолями. Для того, чтобы сама видеоигра правильно функционировала, в ней используются самые распространенные математические вычисления. Чтобы показать, насколько математически основан наш мир компьютерных игр я создала симуляцию для вычисления ряда самых часто использованных функций в видеоиграх.

Цель проекта: показать, где и каким образом используется математика в видеоиграх на примере написанной мною демонстрационной игры.

Задачи проекта:

* Рассчитать расстояние между игроками;
* Найти угол, при котором игрок должен повернуться при движении;
* Вычислить градиент, с которым должен двигаться игрок;
* Рассчитать параболу, по которой игрок будет прыгать;

# **Глава 1 – Начало**

На первом этапе кодаимпортируется библиотеки как «math», «pygame» и «random», которые будут необходимы для того, чтобы работала программа, а также спрашивает несколько вопросов для запуска симуляции и благодаря функции «while» симуляция может повторяться бесконечно, пока ее не отключить (см.Рисунок 2).

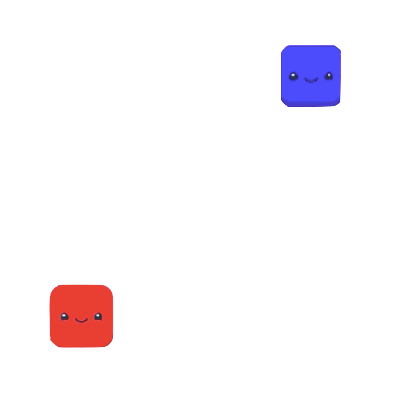
Далее запрашиваются входные данные, чтобы запуститьресурсы «pygame». Для чистоты эксперимента я вызвала функцию «random», которая будет случайно выбирать координаты в промежутке от 0 – 620 для (x, y) – Синего квадрата и для (x2, y2) – Красного квадрата (см. Рисунок 3). Затем изображение игроков будет выведено на отдельном экране 700px на 700px, используя эти случайно выбранные координаты (см. Рисунок 1).

Рисунок1. Персонажи игры



Рисунок2. Начало программы часть 1

Подготовив экран, в следующих этапах будут показаны четыре самые распространенные математические вычисления, которые используются почти в каждой видеоигре.

Рисунок3. Начало программы часть 2



# **Глава 2 – Треугольник между игроками**

Между любыми двумя точками можно построить прямоугольный треугольник, где гипотенуза будет дистанцией между Красным и Синимквадратами. Для вычисления гипотенузы используется теорема Пифагора, но сначала нужно найти стороны треугольника (см.Рисунок4). Стороны треугольника высчитываются в зависимости от того, как расположенны игроки, так как возможны четыре разные ситуации (см.Рисунок 5).

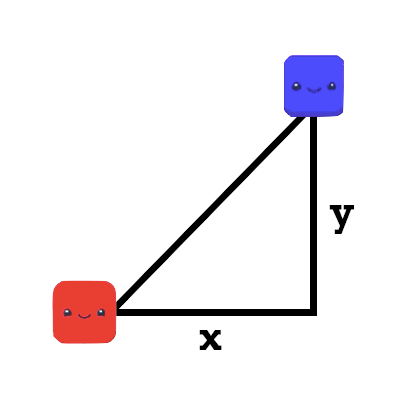
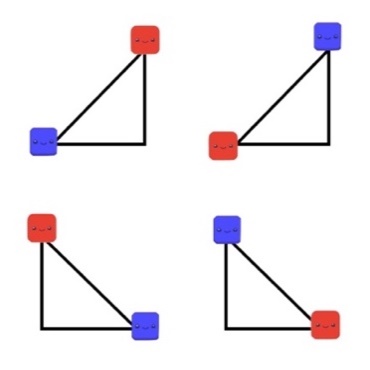


Рисунок 5. Возможные расположения треугольников

Рисунок 4. Треугольник между игроками

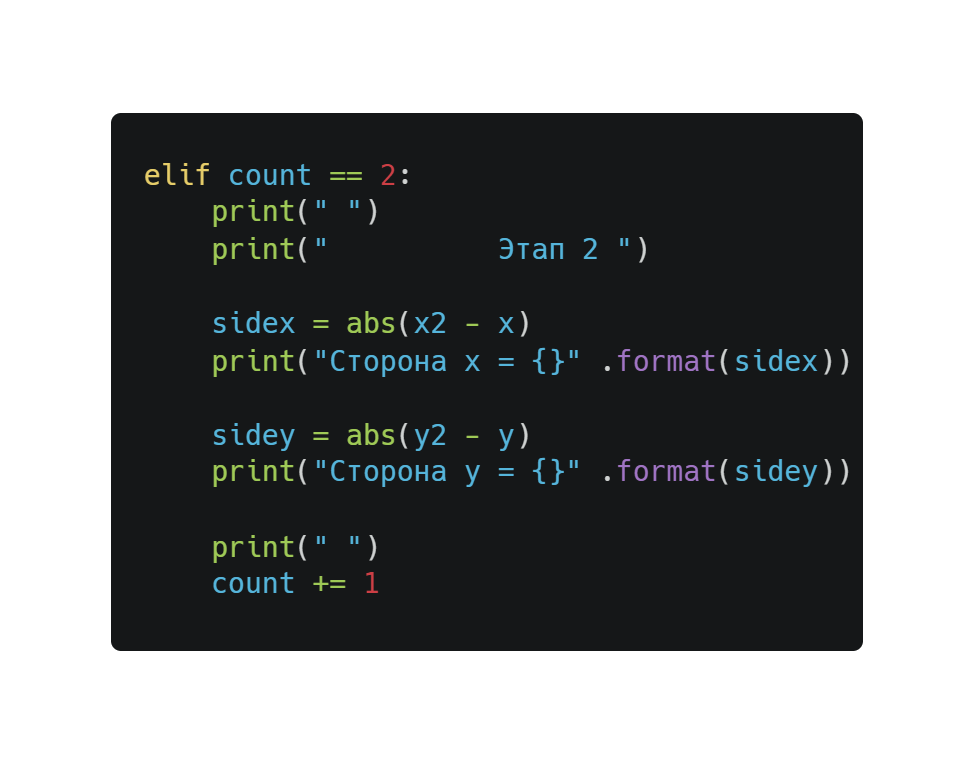
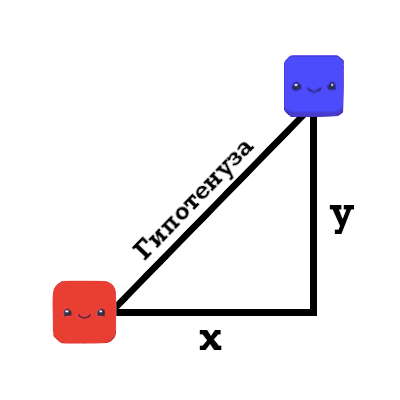
Как можно увидеть в коде, позиции Красного квадрата отнимаются от позицииСинего: (x2 – x) и (y2 – y). Однако, если Синий квадрат имеет более высокие значения, то нужно отнимать в противоположном порядке. Показанный ниже код вычисляет какое из значений X и Y больше и отнимает числа в правильном порядке для того, чтобы мы могли работать с правильными числами в следующих этапах программы (см. Рисунок 6).

Рисунок6. Вычисление сторон X и Y

# **Глава 3– Расстояние между игроками**

****

В результате вычисления двух сторон треугольника мы можем использовать теорему Пифагора для вычисления гипотенузы - расстояния между Красным и Синим квадратами (см. Рисунок 7). Поскольку в библиотеке «math» нет функции «возвести в квадрат», я переписала уравнение в другой форме (Г – Гипотенуза)-

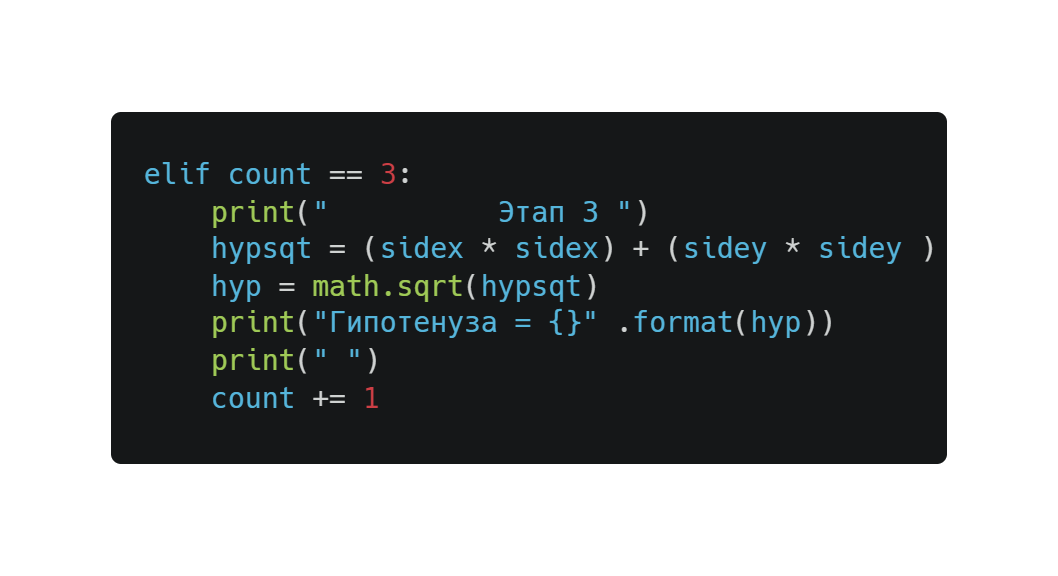
Рисунок7. Гипотенуза треугольника

(X × X) × (Y × Y) = Г × Г

После вычисления Г × Г, используется функция «math.sqrt» для того, чтобы возвести Г × Г в квадратный корень, получить гипотенузу и сохранить в переменной «hyp» (см. Рисунок 8).

Рисунок8. Вычисление гипотенузы

Как пример, это вычисление будет использовано, чтобы показать дистанцию между героями или объектом в игре.



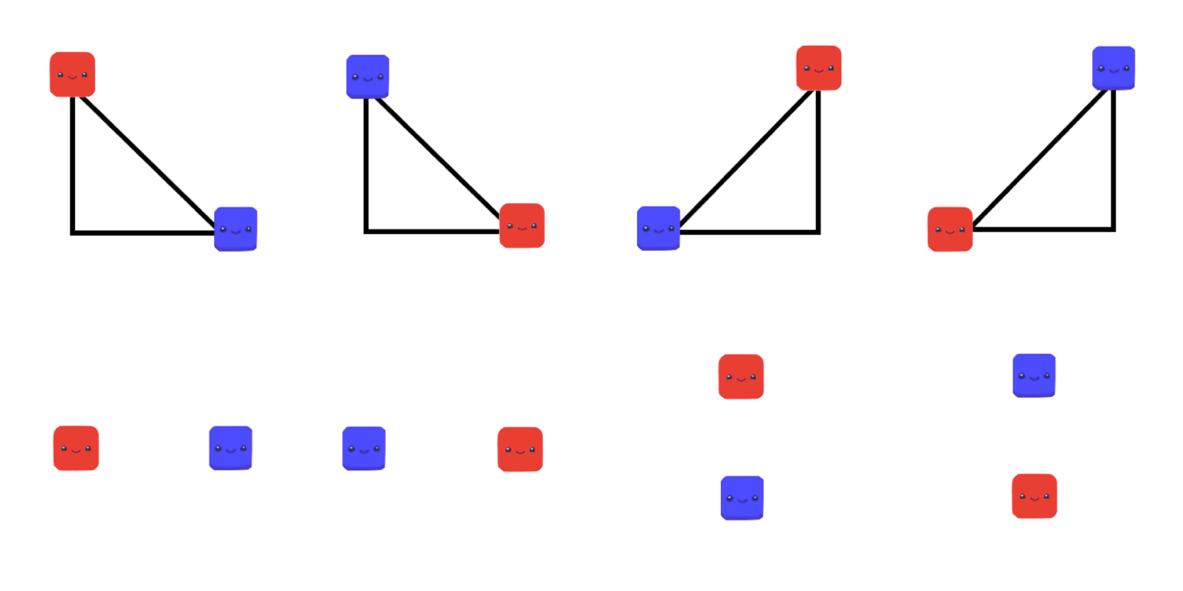
# **../../../../../Downloads/squarw.pngГлава 4 – Угол при повороте**

В четвертом этапе вычисляется угол Красного квадрата по отношению к Синему (см. Рисунок 9) и его пеленг (угол, отсчитываемый с севера, по часовой стрелке) (см. Рисунок 11). В играх это необходимо, чтобы игроки могли поворачиваться лицом к лицу к своим врагам.

Рисунок9. Угол поворота

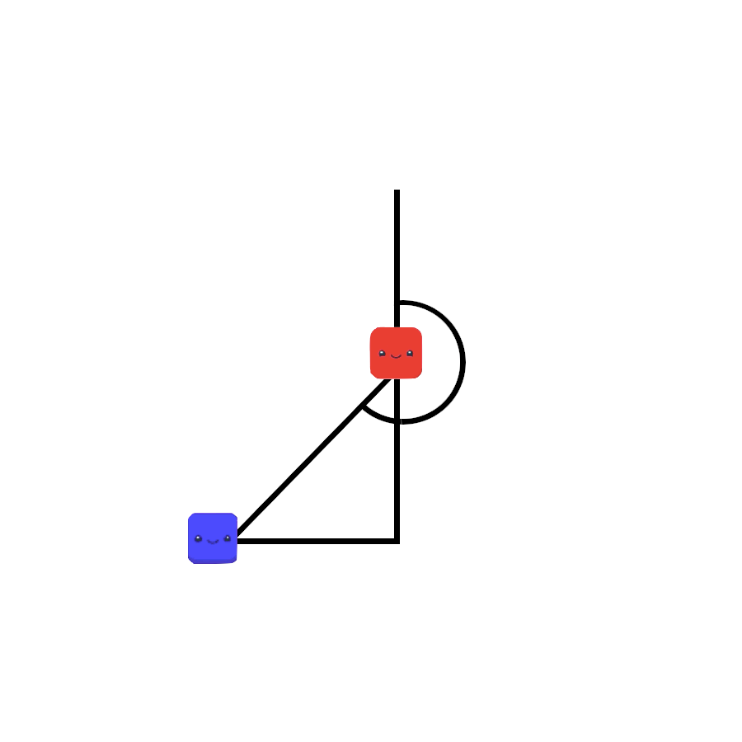
Прежде всего, есть восемь разных расположений игроков

Рисунок10. Возможные расположение игроков

(см.Рисунок 10). В каждом из них углы вычисляются по-разному. А там, где игрокирасположены на одном уровне, между ними нет угла. Это значит, что программа должна определить положение квадратов и соответственно правильно посчитать угол.

В программе я использовала вложенный условный алгоритм для того, чтобы написать ход действий к каждому расположению. Как вы можете увидеть, программа проверяет расположениеквадратов. Затем высчитывает угол прямоугольного треугольника, используя тригонометрию – тангенс (противоположная сторона, разделённая на касательную). Зная угол, можно высчитать пеленг, на который Красный квадрат должен повернуться, чтобы смотреть на свою цель (см.Рисунок 12).

Однако, если квадраты расположены на одной линии, числа X или Y будут совпадать. В зависимости от их расположений пеленг будет равен 0, 90, 180 или 270 градусов.

Например, в этой позиции (см. Рисунок 11), необходимо к углу, высчитанному используя тангенс, добавить 180 градусов. Так как это только демонстрация, то угол поворота высчитывается от севера.

В настоящих играх не всегда игроки расположены относительно севера, поэтому в зависимости от их положения будут добавлены иные градусы.

Рисунок 11. Пеленг игрока

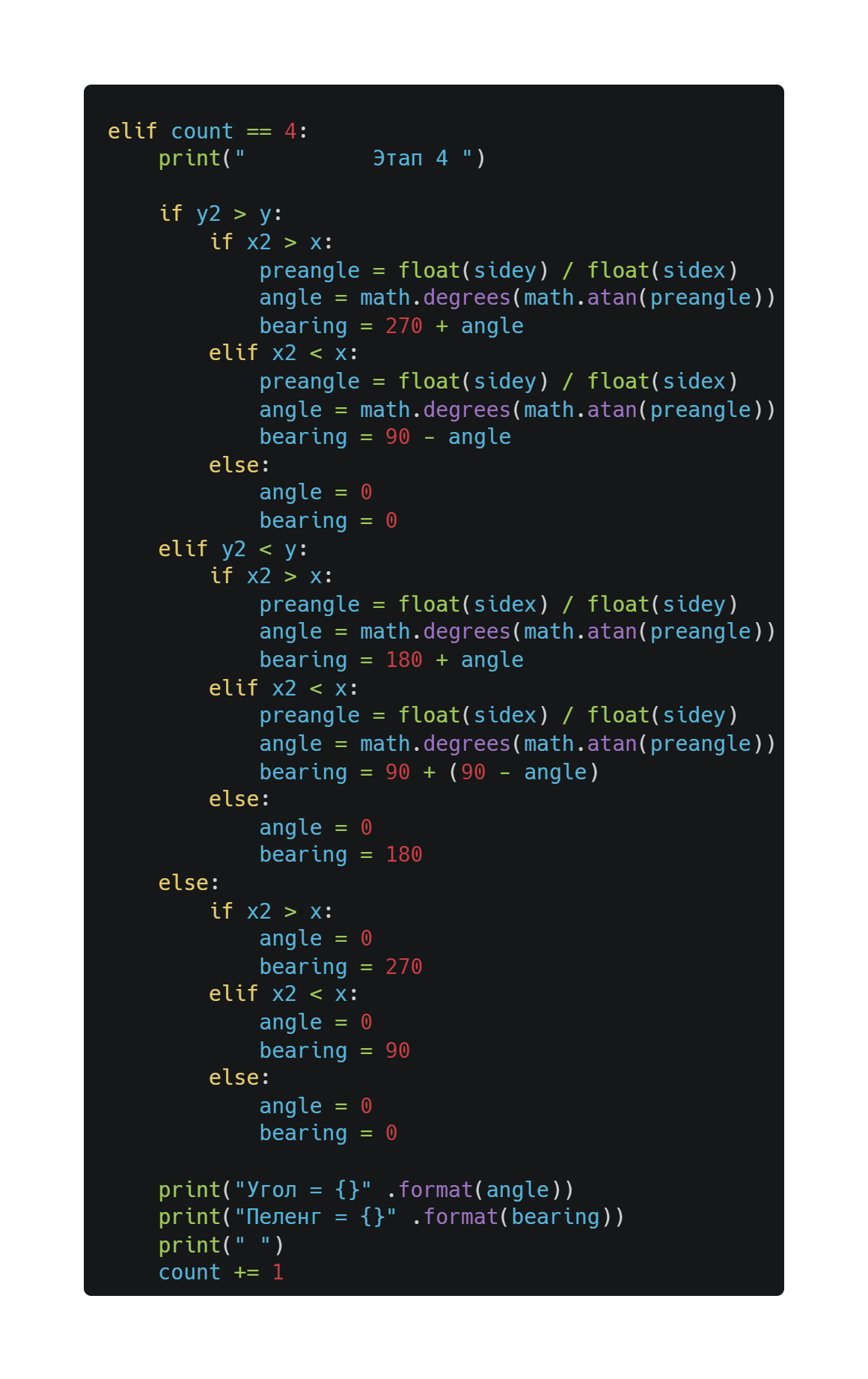


Рисунок12. Вычисления угла и пеленга

# **Глава 5 – Движение по прямой**

## Глава 5.1 - Нахождение градиента

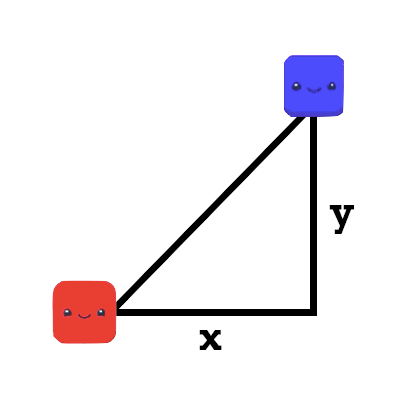
Пятый этап можно разделить на две части. В первом мы находим сам градиент: на сколько нужно переместиться в сторону Y при каждом X (см. Рисунок 13). Благодаря способности компьютеров быстро выполнять вычисления, постоянно печатаются новые картинки и получается эффект движения игрока. Градиент находиться при делении стороны Y на X (см. Рисунок 14).

Рисунок13. Треугольник между игроками

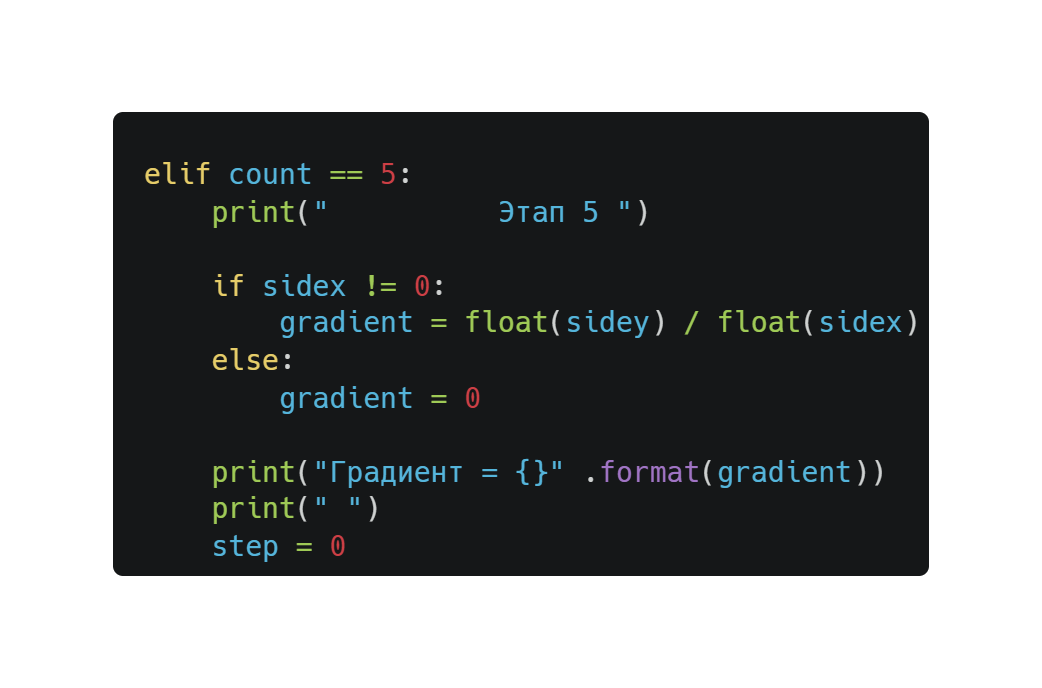


Рисунок14. Вычисления градиента

## Глава 5.2 - Движение игрока

В зависимости от расположения игроков и зная их градиент движения, будет либо прибавляться, либо вычитаться единица от числа «х». А к числу «y» по ситуации будет, либо прибавляться, либо вычитаться градиент, пока числа «x2» и «y2» не станут равны «x» и «y». Однако, если у кубиков одинаковые числа «x» или «y», то программа это выявит и просто будет добавлять единицу к нужному значению (см.Рисунок 15).

Каждый раз, когда меняется какое-то значение, то программа опять напечатает фон, чтобы не было видно предыдущее расположение игроков, затем напечатается новый игрок в его новом положении.

Как я уже сказала выше, благодаря быстрому перемещению картинок, человеческий глаз не замечает, что игрок не двигается, а печатается заново.



Рисунок15. Движение Красного квадрата

# **Глава 6 - Пример из настоящей видеоигры**

В видеоигре «Trigonometry Wars» можно увидеть аналогичные математические вычисления.

Когда враг (Розовый Куб) движется прямо к главному герою, чтобы атаковать его, мы можем нарисовать треугольник, по которому программа будет высчитывать градиент, с котором Розовый Куб должен перемещаться. (см.Рисунок 16)



Рисунок16. Снимок игры «Trigonometry Wars»

# **Глава 7 – Прыжок по параболе**

## Глава 7.1 - Вычисление параболы

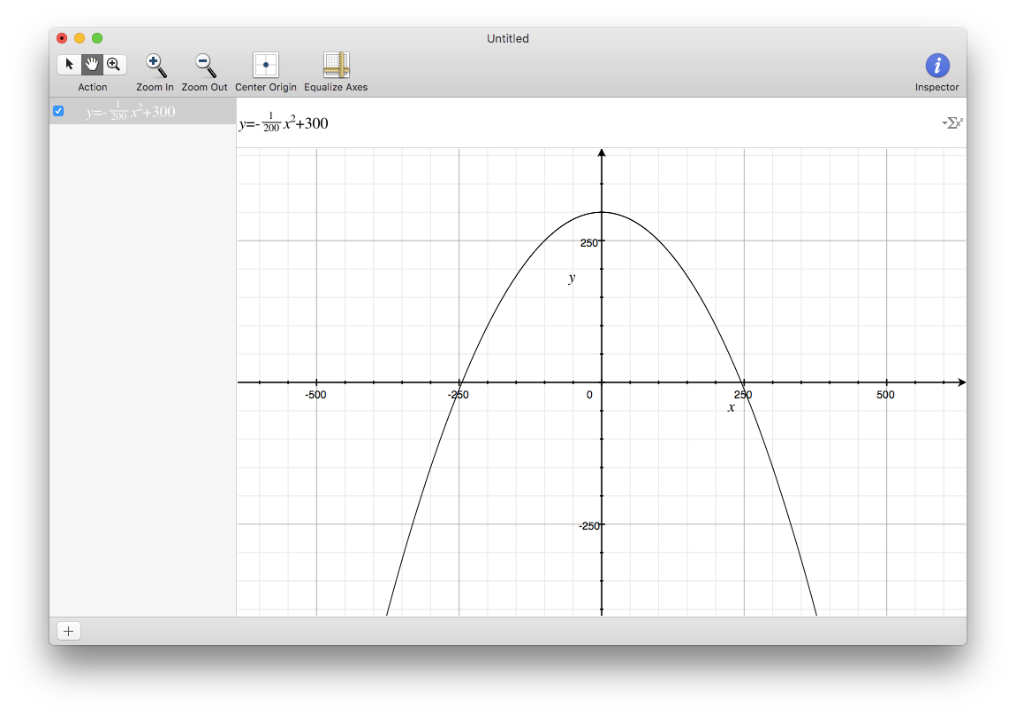
Помимо движения по прямой линии, во многих играх персонажи должны уметь прыгать. Для того чтобы создать данный эффект, подбирается парабола, по которой игрок должен прыгнуть (см. Рисунок 17). В моей демонстрационной игре я использовала параболу с формулой:

Рисунок17. Парабола

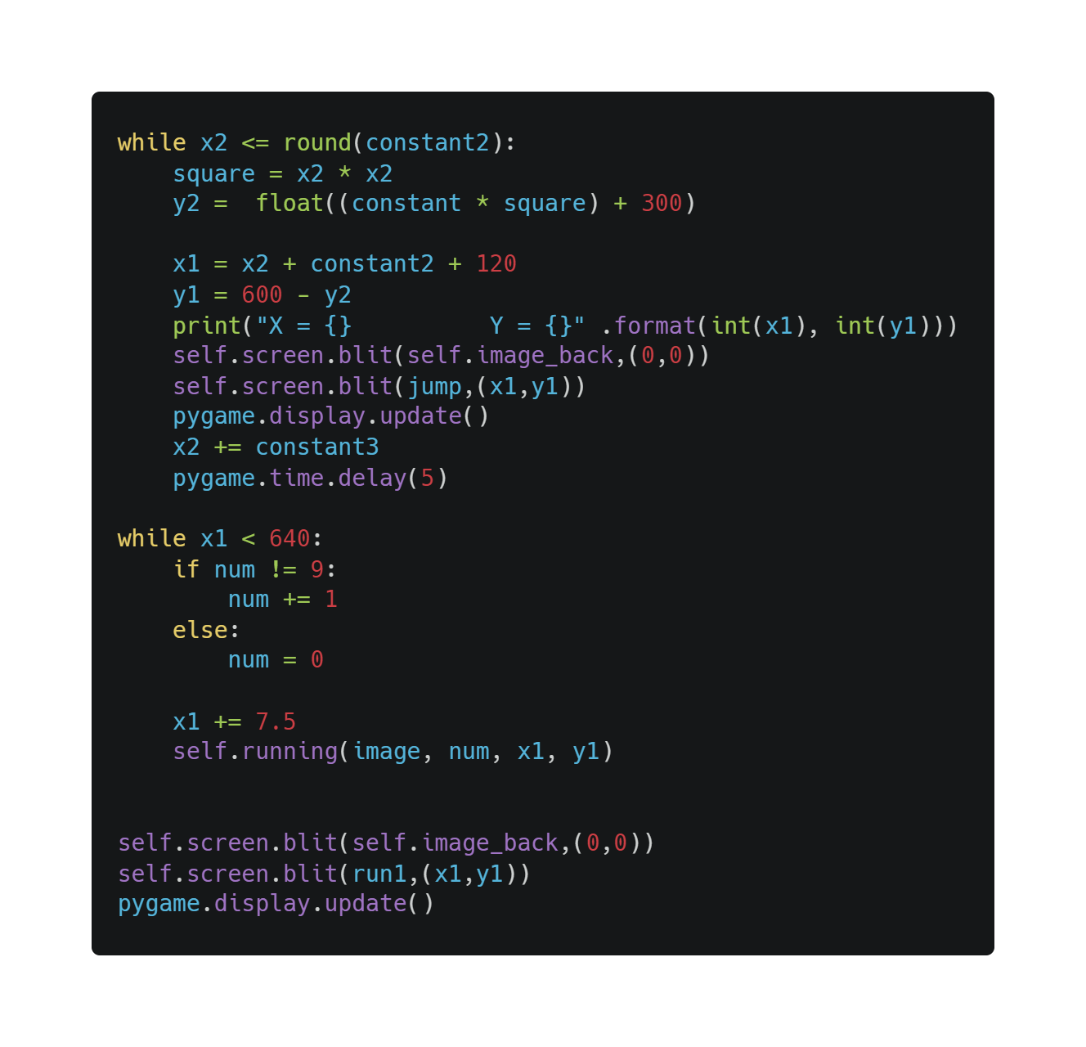
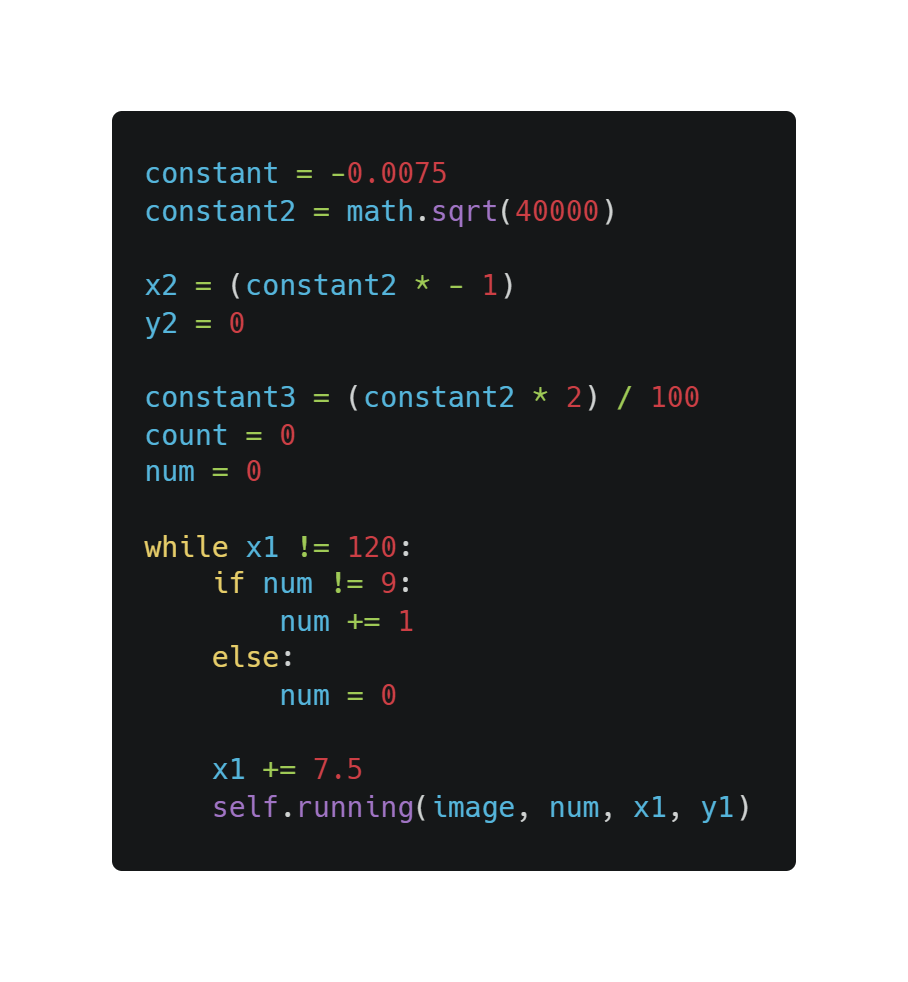
Используя эту формулу параболы, программа будет самостоятельно высчитывать значения «x1» и «y2» игрока, который при изменении своих координат будет вновь печататься на экран. Так как экран начинает счет с нуля и нет значений меньше нуля, то к каждому высчитанному «x1» добавляется некое число, чтобы переместитьпараболу вправо. В коде это число записано в постоянном значении «constant2» (см. Рисунок 18).

Рисунок18. Движение игрока по параболе

## Глава 7.2 - Пример использования параболы

Чтобы более наглядно показать использование параболы в настоящей игре, я создала игрока с десятью разными положениями во время бега и плюс одно прыгающее изображение (см.Рисунок 19). В демонстрационной программе игрок бежит, делает прыжок, после чего он продолжает движение вперед в течении нескольких секунд до полной остановки (см. Рисунок 20).

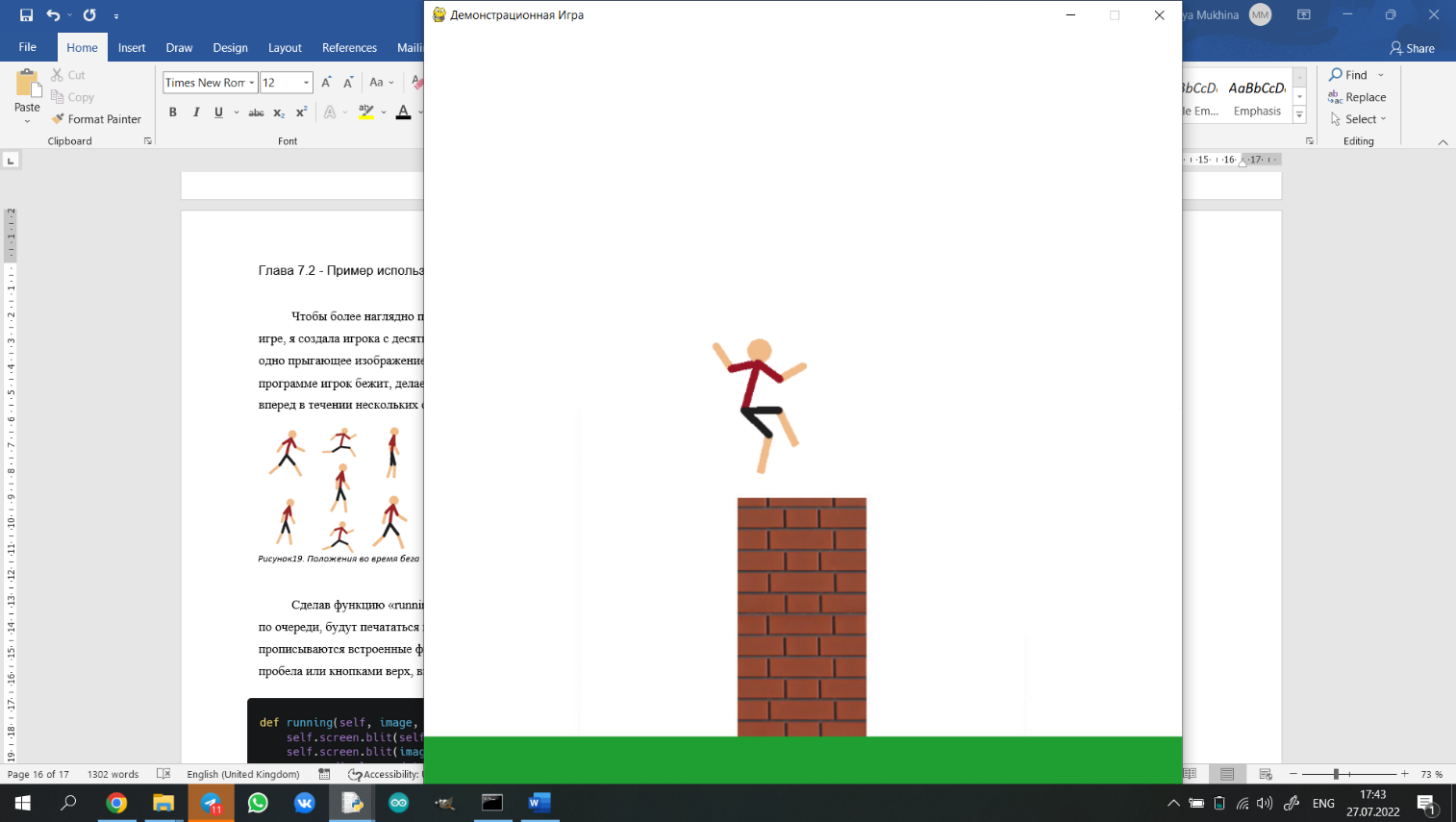
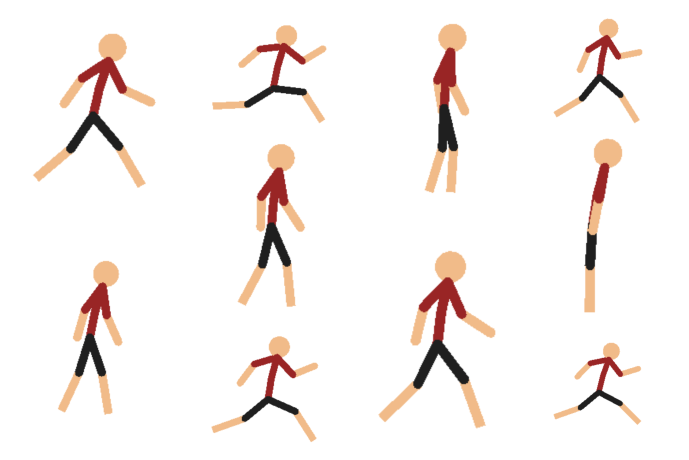
****

Рисунок20. Пример применения параболы

Рисунок19. Положения во время бега

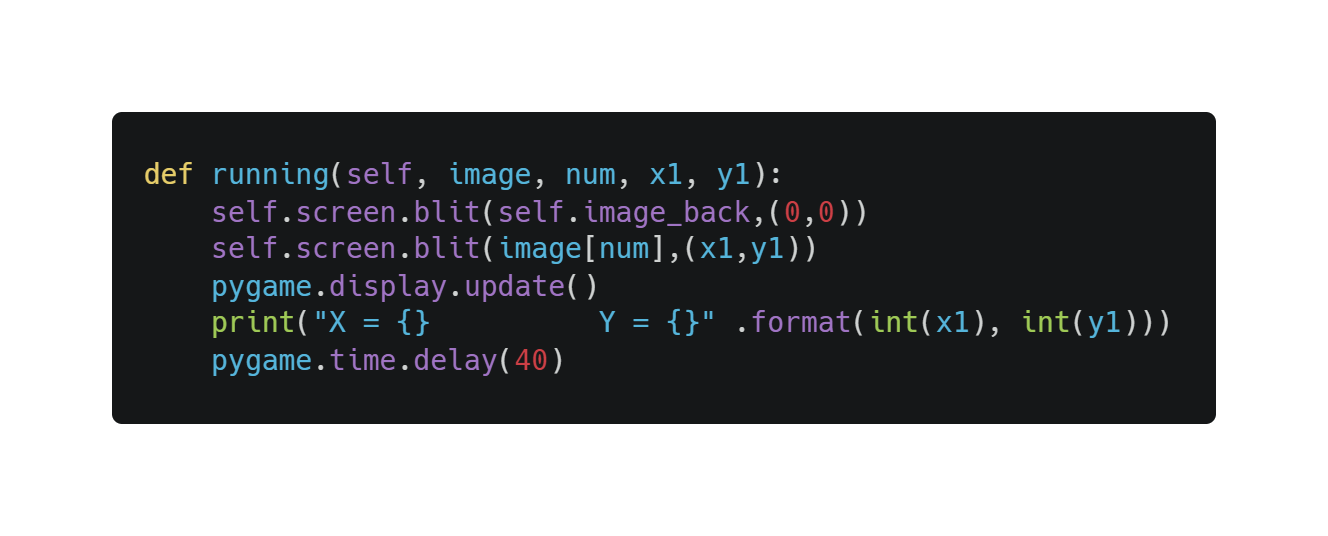
Сделав функцию «running» (см.Рисунок 21), когда она вызывается, то по очереди, будут печататься игроки. Таким образом, в видеоиграх прописываются встроенные функции, которые вызываются при нажатии пробела или кнопками верх, вниз и т.д.

Рисунок21. Функция «running»

# **Заключение**

При помощи демонстрационной игры я показала, что даже самые простейшие видеоигры основаны на математике. В этом проекте было разобрано как при помощи самых простых математических вычислений мы можем:

* рассчитатьрасстояние между игроками
* найти угол, при котором игрок должен поворачиваться при движении
* высчитать градиент, с которым должен двигаться игрок
* рассчитать параболу, по которой игрок будет прыгать