|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт перспективных технологий и индустриального программирования (ИПТИП)**

**Кафедра индустриального программирования**

**Отчёт по практическому заданию № 1**

**Тема практического задания**: Создать приложение с использованием динамических объектов подобную игре «Арканоид».

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет представлен к рассмотрению Студентом группы ЭФБО-02-22 |  | Кондрашовым Денисом Михайловичем | |
|  |  | |  | |  |

Москва 2024 г.

В качестве языка программирования был выбран Python, а в качестве графической библиотеки был выбран Pygame.

Для начала импортируем все нужные библиотеки:

Изображение выглядит как текст, Шрифт, белый

Автоматически созданное описание

Был импортирован Pygame и Math, а также взята функция randint из библиотеки Random.

Далее, нужно инициализировать переменные:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

В данном случаи инициализируется библиотека Pygame. Затем задаются переменные для ширины и высоты окна. После них идёт объявление переменных для количества шаров и их максимальной начальной скорости.

Затем создаётся окно Pygame с заданными параметрами. Окну присевается название «Арканоид».

Далее задаётся переменная FPS для скорости основного игрового цикла. После чего курсор устанавливается в видимое положение и инициализируется внутриигровое время.

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Далее идут загрузки спрайтов. Спрайты находятся в той же папке, что и код. Заметьте, что некоторые спрайты имеют транспарентность по определённому цветовому ключу.



Последние две переменные отвечают за нажатие кнопки запуска и работу окна. Далее идёт основной игровой цикл:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Цикл работает до тех пор, пока переменная «run» по каким-либо причинам не станет ложью. Первое, что происходит в самом цикле, это устанавливается частота кадров.

Без данной команды код будет выполняться с максимально доступной скоростью. Это значит, что при изменении производительности (например, запуске кода на другом устройстве или при открытии других программ) будет изменяться скорость работы программы. Это может быть очень неудобно, так как невозможно будет предсказать скорость работы программы в других условиях.

Наличие данной команды ограничивает скорость прохода цикла до 60 раз в секунду (или 1 раз в примерно 0.016 секунд). Если код может выполниться быстрее, то данная команда остановит его выполнение до тех пор, пока не пройдёт 1/60 секунды.

Далее идёт получение координаты курсора в рамках окна.

Затем идёт цикл обработки событий. Данный цикл обрабатывает все возникшие события за время одного прохода цикла. Это могут быть такие события как нажатие клавиши, передвижение курсора или другие. Цикл обработает каждое такое событие отдельно. Если события не произошли – цикл не запустится.

Цикл обработки событий проверяет событие и если оно сходится с чем-либо, то выполняет соответствующую команду.

Если событие – нажатие кнопки «Закрыть», то переменная «run» станет ложью.

Если событие – нажатие клавиши и нажатая клавиша – Escape, то переменная «run» станет ложью.

Если была нажата клавиша мыши и код нажатой клавиши – 1 (что соответствует левой кнопки мыши), то код смотрит, где были координаты курсора, когда была нажата клавиша мыши. Если координаты курсора попадают по кнопке «Начать», и кнопка ещё не была нажата, то происходит запуск игры.

Изображение выглядит как текст, Шрифт, белый, дизайн

Автоматически созданное описание

В финальной части кода можно увидеть вызов функции Update(), которая производит обновление экрана. После того, как программа была закрыта, произойдёт выход из Pygame.



Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Функция обновления экрана вызывается каждый кадр для обновления содержимого на экране. Сначала она обновит весь экран, залив его одним цветом. Затем, если кнопка старта ещё не была нажата, то она будет отображена на экране.

После завершения всех отрисовок, идёт рендер содержимого на экран.

Имея базовую программу, можно реализовать логику подвижных шариков. Для этого создадим соответствующий класс:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Данный класс имеет переменные для положения шарика в пространстве, его радиуса и его скорости. Эти переменные можно задать как при создании шарика, так и после с помощью специального метода.

Метод update() обновит положение и скорость шарика. Для этого он сначала добавит к текущим координатам скорость. Заметьте, что скорость умножается на 1/60, так как цикл идёт именно с этой скоростью. Это также означает, что скорость задаётся в пикселях в секунду.

Затем идут проверки на пересечение шарика с границами окна. В случаи, если граница была нарушена, скорость шарика будет инвестирована.

Далее, нам нужна функция для проверки пересечения двух шариков. Окружность, будучи формой без точек как таковых, не может иметь привычную коллизию. Всё что у нас есть это центр окружности и его радиус. Имея эти вводные, можно понять, входит ли какая-нибуть точка пространства в окружность или нет, используя Теорему Пифагора для нахождения расстояния между точкой и центром окружности. Узнав это расстояние, мы можем сравнить его с радиусом, и если радиус оказался больше, то значит точка входит в окружность. В противном случаи точка находится за пределами окружности.

Но что нужно сделать, чтобы понять, пересекаются ли две окружности? Если две окружности соприкасаются границами, то расстояние от центра одной окружности до центра другой будет суммой их радиусов. Это значит, что если мы измерим расстояние от центра одной окружности до другой и оно окажется меньше, чем сумма их радиусов, то можно сказать, что окружности пересекаются. В противном случаи пересечения нет.

Изображение выглядит как текст, Шрифт, снимок экрана, белый

Автоматически созданное описание

Данная функция является ключевой, позволяя определять столкновения шаров в пространстве.

Впрочем, данную функцию нужно усовершенствовать, чтобы она могла не только проверять пересечение окружностей, но и изменять их параметры при столкновении.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

В данной функции, как и в прошлой проверяется пересечение шаров. Функция не только проверят пересечение, но и то, как сильно шары пересекаются друг с другом.

Затем, ищется нормали столкновения, то есть то, как сильно столкновение повлияло на векторы X и Y.

Чтобы шары после окончания столкновения не застряли друг в друге, функция отодвигает шары друг от друга на расстояние, при котором они не будут пересекаться.

Наконец, с помощью нормали столкновения и текущей скорости шаров высчитывается и задаётся скорость шаров после столкновения.

Такая функция позволяет изменять параметры шаров при столкновении.

Теперь нужно задать массив с самими шарами:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Так как мы хотим иметь несколько типов шаров, то можно просто сделать два массива, которые не будут взаимодействовать друг с другом, но при этом взаимодействовать внутри себя. Это позволит шарам одного типа взаимодействовать с шарами такого же типа и игнорировать шары другого типа.

Для каждого массива генерируются шары с различными параметрами. Случайно выбираются начальные координаты шаров и их скорость. Радиус шаров остаётся неизменным, но при желании его также можно изменить или сделать случайно генерируемым (но в таком случаи спрайт шара может не сходиться с его математическими размерами).

Далее, в основной игровой цикл добавим код для обработки шаров:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Данная часть кода будет работать только в том случаи, если была нажата кнопка запуска. Каждый шар обновляет свою позицию в соответствии с текущей скоростью и положением. Затем, шар проверяет столкновения с другими шарами и обновляет свои параметры и параметры другого шара в соответствии с функцией обработки столкновений. Заметьте, что изменение порядка подачи шаров в функцию не имеет значения, поэтому нету смысла обрабатывать столкновение первого шара со вторым, если было обработано столкновение второго шара с первым.

Добавим код для отображения шаров:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Теперь функция обновления экрана отобразит шары на их текущей позиции. Так как отображение шаров не зависит от переменной запуска игры, шары будут отображены даже до нажатия кнопки старта. Заметьте, что второй массив находится в обработке после первого, что значит, что шары второго типа будут отрисованы поверх шаров первого типа.

Наконец, можно запустить программу и посмотреть на результат:

Изображение выглядит как снимок экрана, Красочность

Автоматически созданное описание

Изначально все шары не подвижны, так как они не будут обработаны до тех пор, пока не будет запущена игра. Но уже на этом этапе все шары имеют своё положение и скорость.

Нажав на кнопку, мы запустим игру:

Изображение выглядит как снимок экрана, Красочность, Цвет Majorelle blue, синий

Автоматически созданное описание

Кнопка запуска пропала, а шары начали хаотично двигаться. Шары сталкиваются друг с другом, изменяя свою скорость и положение. Красные шары не сталкиваются с синими шарами, так как эти шары принадлежат к разным типам. А шары синего цвета обрисовываются поверх красных, так как идут позже в функции обновления экрана.