Отчет по лабораторной работе 1

Студента группы №3331506/00401 Орехова Алексея Михайловича

Код работы выложен на [гитхаб](https://github.com/alexeyorehoff/cv-lessons)

В рамках работы №1 было выполнено два задания – базовое и дополнительное задание №2 – Задача трех тел. С целью унификации задания 1 и 2 были созданы два вспомогательных класса:

* Класс App – объединяющий в себе создание окна, хранение, анимацию движения и отрисовку спрайтов.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

* Класс Sprite – класс, хранящий в себе текстуру и альфа-маску для вывода изображений с прозрачным фоном. Также, класс позволяет задавать положение спрайта на экране как через прямой ввод координат, так и через функцию анимации параметрически.

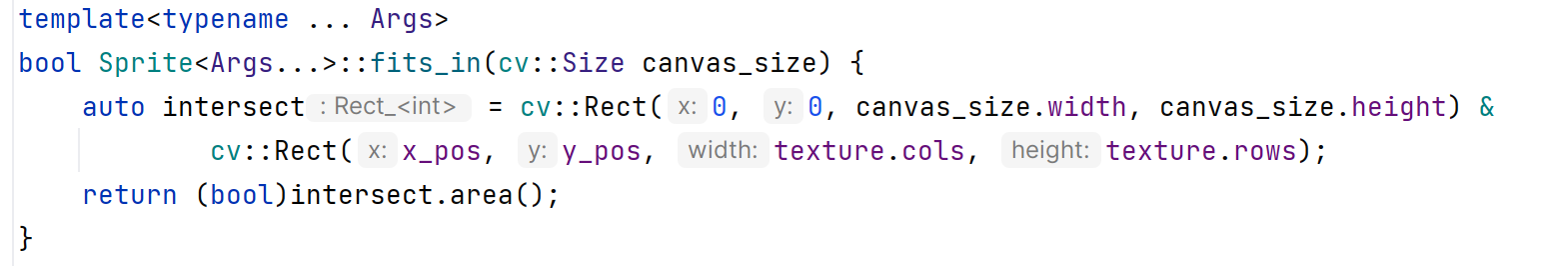
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Аргументы, которые принимает функция анимации вводятся как шаблон. В конструкторе базового класса Sprite он принимает путь к текстуре в формате .png и производит разложение картинки на альфа маску и текстуру.

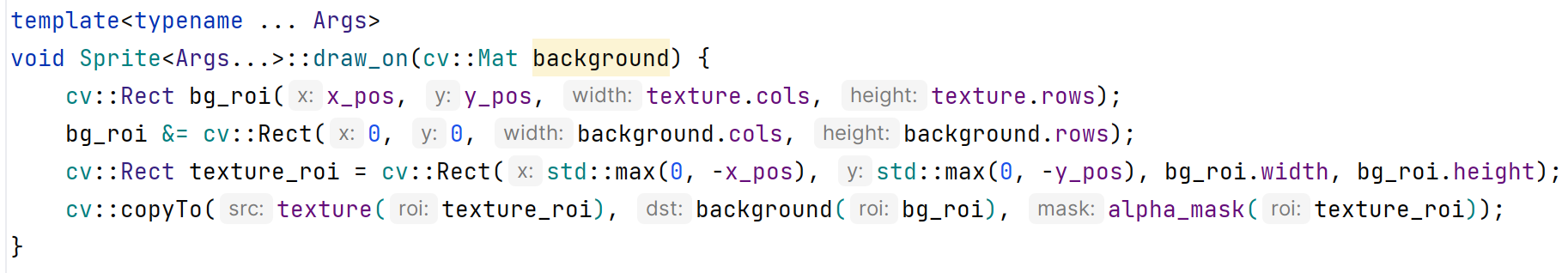
Из соображений удобства, со стороны пользователя ноль координат спрайта находится не в левом верхнем углу, а в центре. Для этого введен метод set\_position, производящий преобразование.

В случае задания позиции через функцию анимации пользователь задает анимирующую функцию и после в каждой итерации цикла отрисовки с помощью метода animate передает параметры, после чего положение спрайта обновляется.

Для отрисовки спрайта на фоне используются два метода. Метод fits\_in проверяет, попадают ли хоть какие-то пиксели спрайта в пределы экрана:  


С его помощью метод отрисовки спрайтов класса App, который будет рассмотрен позже, выбирает, попадают ли спрайты на экран.

Метод draw\_on копирует текстуру спрайта на фон, при этом, из-за того, что выбор области интереса за пределами изображения вызовет ошибку, этот метод обрезает ROI по границам экрана, после чего формирует ROI для текстуры и маски чтобы размер ROI на фоне и на текстуре были равны.



Класс App содержит в себе хранилище для хранения спрайтов в виде std::vector и матрицу для хранения оригинала фонового изображения. По умолчанию, пользователь в главном цикле отрисовки может вызвать функцию render, которая пройдется по всем спрайтам хранилища и вернет фоновое изображение с отображенными на нем спрайтами в соответствии с правилом, что отрисовываются только спрайты, где хоть один пиксель попадает на область экрана.



В основной программе создан класс SimpleSpriteApp, унаследованный от созданного ранее класса App с подставленным шаблоном Sprite<int>. Типы данных в шаблоне класса спрайт означает количество и типы аргументов, используемые для анимации спрайта. В данном случае задается только параметр координаты по оси Икс типа int. Поле speed соответствует приращению координаты за один такт отрисовки, первоначально он равен 1, что соответствует скорости 1 пиксель за такт в положительном направлении. Перегружается метод render родительского класса для задания поведения спрайтов. Так, каждый такт к координате прибавляется «скорость», для каждого спрайта на экране вызывается метод animate, пересчитывающий их положение. На новом положении спрайтов на исходном изображении рисуется круг, отображающий траекторию спрайтов на экране. В конце вызывается метод render родительского класса, отрисовывающий сами спрайты. Также, при пересечении спрайтом границ экрана, его скорость инвертируется.



В функции main остается создать объект класса SimpleSpriteApp, добавить на него спрайты, для которых будет задана функция анимации, а после в цикле отображать отрисованные кадры на экране. В момент пересечения спрайтами центра экрана делается снимок экрана, а при нажатии на клавишу Esc клавиатуры, происходит выход из программы.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Результаты работы программы показаны в приложенном видео.

Для второго задания были использованы базовые классы, описанные ранее. Для тел был унаследован класс Body. Кроме базовых полей, в них были добавлены данные о массе тела и составляющих скорости.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание В классе NBodyApp в функции render был создан алгоритм, позволяющий рассчитывать равнодействующую сил притяжения от других тел на экране для каждого из тел, скорость тела изменялась в соответствии с вычисленным ускорением за промежуток времени, который задавался локальной переменной dt. С ее помощью можно управлять скоростью работы симуляции. Далее позиция каждого из тел пересчитывалась в соответствии с расстоянием, пройденным ими за все тот же промежуток времени.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Аналогично предыдущему заданию, в главной функции были созданы объекты тел с определенными значениями масс, скоростей и начальных положений и отрисованы в цикле.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Результаты работы программы, созданной в задании 2 представлены в приложенном видео.