Курсовая робота

студента группы КН-21 Тесленко Ильи

Разделы:

1. Введение
2. Моделирование движения объектов на компьютере

2.1. Основные подходы к моделированию движения объектов

2.2 Алгоритм моделирования движения

2.3 Использование движка визуализации PIXI

1. Описание разработанного класса SpawnerFood
2. Описание класса Food
3. Использование класса SpawnerFood вместе с движком визуализации PIXI
4. Выводы, эксперимент и возможные улучшения класса
5. Список использованных источников

Раздел 1: **Введение**

Современные компьютерные игры и приложения многократно расширили возможности компьютеров и смартфонов, позволяя пользователям получать удовольствие от виртуального взаимодействия с окружающими объектами. Один из ключевых аспектов, который делает такое взаимодействие увлекательным и интересным, это движение объектов на экране. Моделирование движения объектов на компьютере важно не только для создания игр, но также и для разработки роботов, управляемых компьютером, а также для исследования физики.

Одним из способов моделирования движения объектов является создание алгоритмов, которые задают направление и скорость движения каждого объекта на основе его текущего состояния и окружающей среды. Такие алгоритмы часто называют "боидами" ("boids"), потому что они напоминают поведение стаи птиц или рыб в природе. Боиды могут использоваться для моделирования движения воды, огня, многих животных и даже пешеходов в городской среде.

Цель данного курсового проекта - создание программы, которая будет моделировать движение объектов на экране с помощью алгоритма боидов. Основными задачами проекта являются:

* разработка алгоритма боидов для моделирования движения объектов;
* создание объектов на экране, которые будут двигаться с помощью разработанного алгоритма;
* визуализация движения объектов на экране с помощью движка визуализации PIXI.
* Для решения этих задач будет использоваться язык программирования JavaScript и библиотека PIXI для визуализации. Результатом проекта будет программа, которая позволит пользователю создавать различные объекты и наблюдать за их движением на экране с помощью алгоритма боидов.

Раздел 2: **Моделирование движения объектов на компьютере**

2.1 **Основные подходы к моделированию движения объектов**

Моделирование движения объектов на компьютере является широкой областью исследований и разработок. Одним из самых популярных подходов является моделирование поведения объектов с помощью алгоритмов boid. Алгоритм boid был разработан Крэйгом Рейнольдсом в 1986 году. Этот алгоритм моделирует поведение стай птиц или рыб, где каждое животное движется в определенном направлении и скорости, а также учитывает поведение окружающих животных.

Существует несколько базовых правил для объектов, моделирующих стай птиц или рыб, которые позволяют им двигаться вместе без столкновений. Первое правило заключается в том, что каждый объект движется в направлении среднего положения ближайших объектов. Второе правило заключается в том, что каждый объект движется в направлении средней скорости ближайших объектов. И, наконец, третье правило заключается в том, что каждый объект движется, чтобы избегать столкновений с ближайшими объектами.

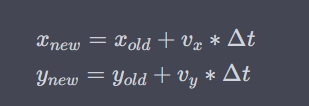
Кроме алгоритма boid, существует множество других подходов к моделированию движения объектов на компьютере. Например, алгоритмы моделирования движения жидкостей, где объекты движутся, смешиваясь и перемещаясь вместе с другими объектами в жидкости. Также существуют подходы, основанные на физических законах, которые моделируют движение объектов, учитывая силы, действующие на них. Эти подходы могут быть использованы для моделирования движения твердых тел, жидкостей, газов и т.д..

2.2 **Алгоритм моделирования движения**

Алгоритм моделирования движения объектов на экране, как правило, сводится к определению их текущей скорости и позиции на основании имеющихся данных о предыдущем положении и направлении движения. Для этого используются математические вычисления, основанные на принципах кинематики и динамики тел.

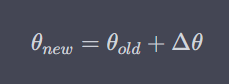
В контексте нашего проекта, класс Food, который создает новые объекты еды, не двигается сам по себе, но его создаваемые объекты должны иметь случайную скорость и направление движения. Для этого в классе Food был реализован метод move, который изменяет координаты создаваемых объектов в соответствии с их скоростью и направлением.

Метод move в классе Food использует следующие математические формулы для вычисления новой позиции объекта на основе его предыдущей позиции, скорости и направления движения:



где $x\_{old}$ и $y\_{old}$ - координаты объекта на предыдущем шаге, $v\_x$ и $v\_y$ - его скорость по горизонтали и вертикали соответственно, $\Delta t$ - время, прошедшее с предыдущего шага.

Для определения направления движения объекта, используется случайное значение угла поворота относительно его текущего направления:



где $\theta\_{old}$ - угол направления объекта на предыдущем шаге, $\Delta \theta$ - случайное значение угла поворота.

Таким образом, метод move позволяет создаваемым объектам еды двигаться со случайной скоростью и направлением, создавая при этом эффект реалистичного движения.

2.3 **Использование движка визуализации PIXI**

Для визуализации создаваемых объектов еды был использован движок PIXI. PIXI – этo библиотека на языке JavaScript для создания интерактивных графических приложений, основанная на WebGL и Canvas.

Для использования PIXI необходимо создать экземпляр класса PIXI.Application, который представляет собой контейнер для отображения графики на экране. Затем создаются объекты класса PIXI.Graphics, которые представляют собой графические примитивы, такие как прямоугольники, круги, полигоны и многое другое. Каждый графический примитив создается путем вызова соответствующего метода класса PIXI.Graphics, например, метода drawRect() для создания прямоугольника или метода drawCircle() для создания круга.

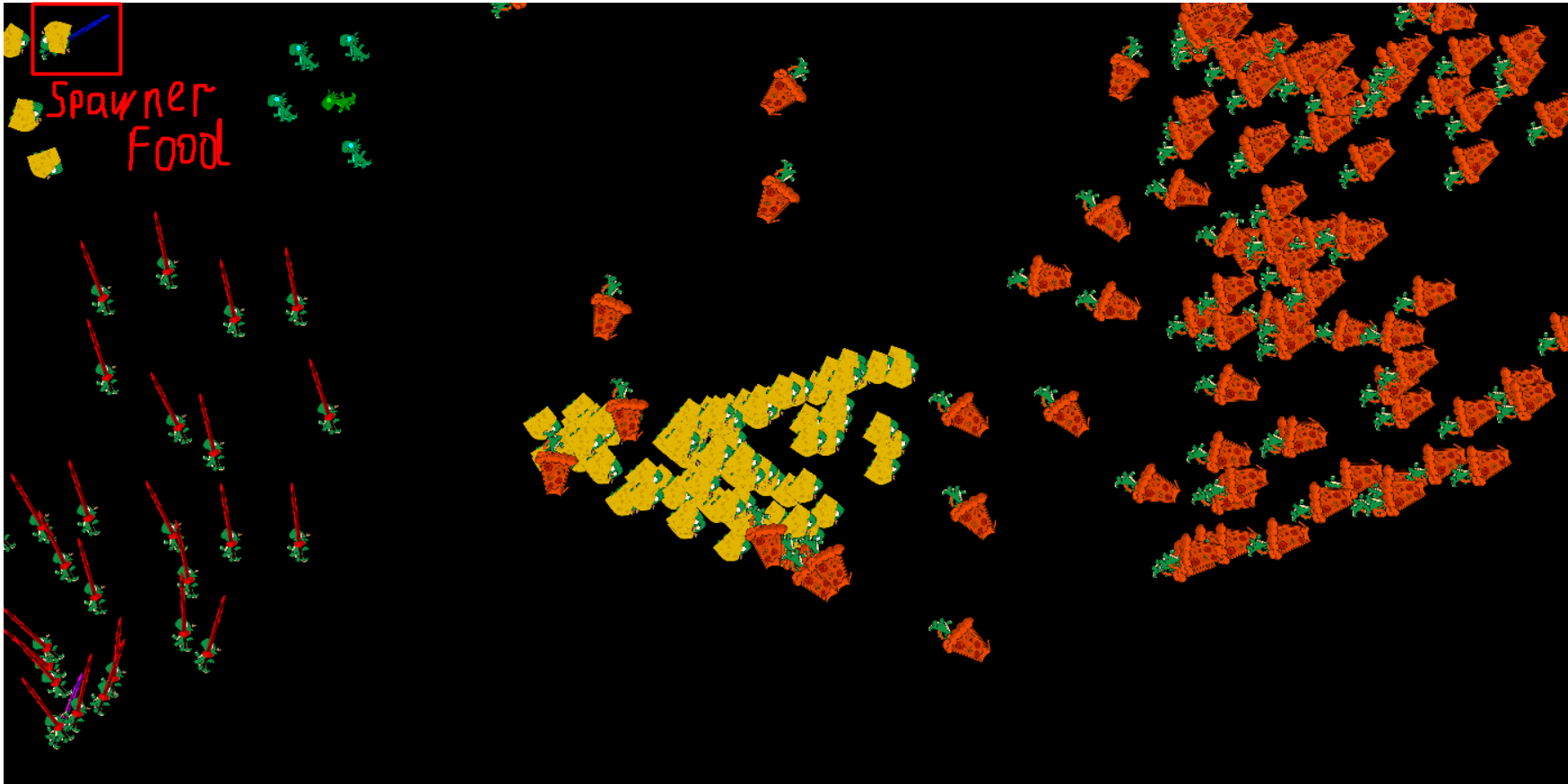
После того, как графические примитивы созданы, их можно добавить на сцену с помощью метода addChild() класса PIXI.Container. Контейнеры представляют собой объекты, которые могут содержать другие объекты, включая другие контейнеры. Для создания объекта контейнера можно использовать класс PIXI.Container.

PIXI также предоставляет множество других классов и методов для работы с графикой, таких как класс PIXI.Sprite для отображения спрайтов и класс PIXI.Text для отображения текста. Кроме того, с помощью PIXI можно загружать и использовать изображения, видео и другие ресурсы.

В целом, PIXI представляет собой мощный инструмент для создания игр и интерактивных приложений с использованием JavaScript и HTML5. Он обладает широким набором функций и поддерживается активным сообществом разработчиков, что делает его привлекательным выбором для многих проектов.

Раздел 3. **Описание разработанного класса SpawnerFood**

Класс SpawnerFood расширяет класс Enemy и представляет собой особый тип вражеского объекта, который создает объекты еды в игре.



Вот подробное описание того, что делает этот класс:

**Конструктор**:

class SpawnerFood extends Enemy {

                    constructor(x,y){

                        super(x,y);

                        this.speed = 0.1;

                        this.image.tint = 0x000fff ;

                        this.timePassed = 0;

                        this.lastCreationTime = 0

                    }

Конструктор вызывается при создании нового объекта SpawnerFood. Он принимает координаты x и y в качестве параметров, чтобы установить начальное положение объекта.

Внутри конструктора вызывается метод super(x, y), который вызывает конструктор класса Enemy и устанавливает начальное положение объекта SpawnerFood.

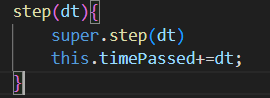
Затем устанавливается свойство speed объекта SpawnerFood равным 0.1, указывая его скорость перемещения.

Свойство tint изображения объекта SpawnerFood устанавливается в 0x000fff, придавая ему синий цвет.

Инициализируется свойство timePassed значением 0, чтобы отслеживать время, прошедшее с момента создания объекта.

Свойство lastCreationTime устанавливается в 0 и будет использоваться для определения времени создания нового объекта еды.

Метод step(dt):



Метод step вызывается на каждом шаге обновления игры и принимает параметр dt, который представляет время, прошедшее с последнего обновления.

Внутри метода вызывается метод super.step(dt), который обрабатывает обновление позиции объекта на основе его скорости и времени.

Свойство timePassed увеличивается на значение dt, чтобы отслеживать прошедшее время.

Метод makeDecision():

Метод makeDecision вызывается для принятия решений объектом SpawnerFood.

Сначала вызывается метод super.makeDecision(), который позволяет объекту Enemy принять свое решение.

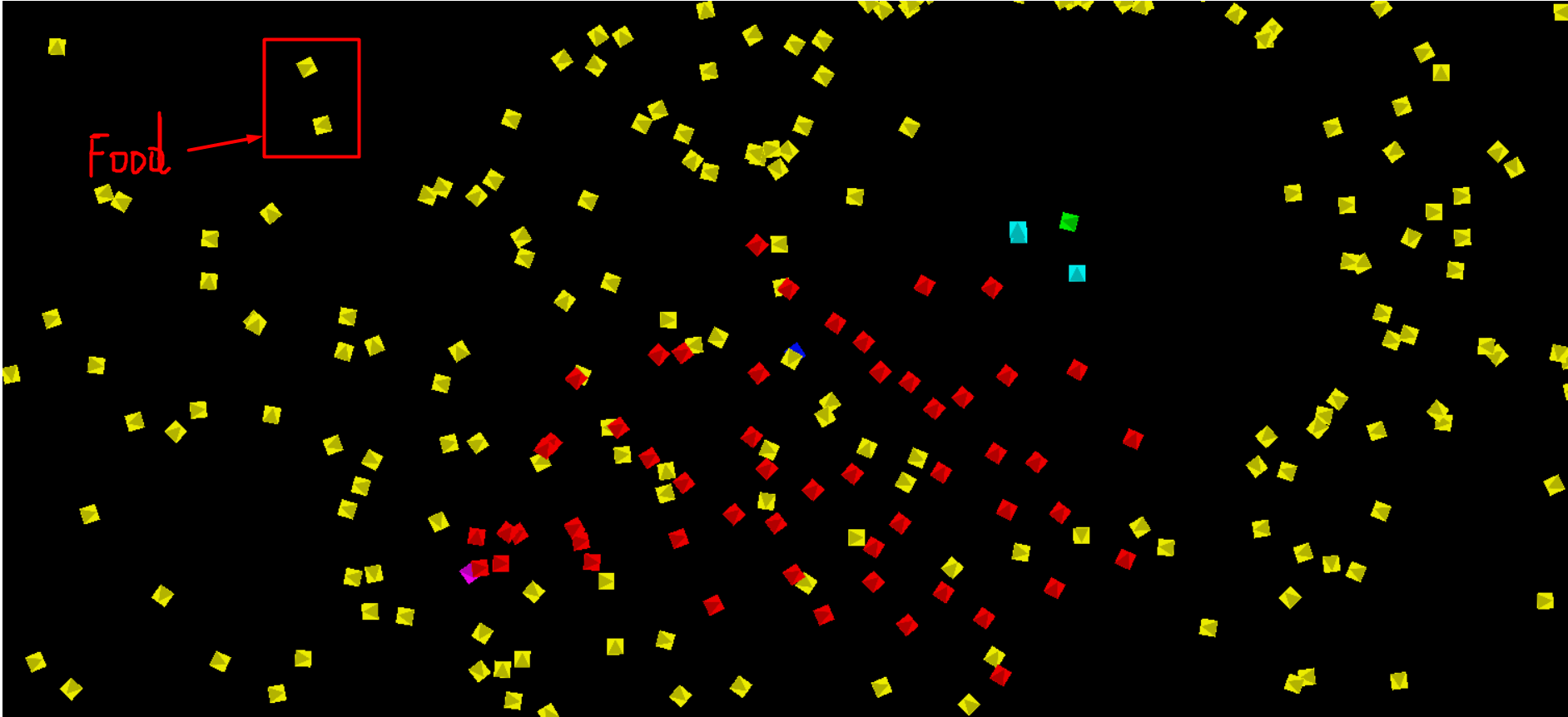
Затем проверяется, прошло ли достаточно времени (больше или равно 50) с момента последнего создания объекта еды. Если условие выполняется, вызывается метод world.createFood(this.x, this.y), чтобы создать новый объект еды в позиции SpawnerFood и lastCreationTime обновляется до текущего значения timePassed.

Затем вычисляются расстояния dx и dy между объектом SpawnerFood и игроком.

Если расстояние d между ними меньше или равно 250, вызывается метод `move2Point(this.x-dx, this.y-dy)

Раздел 4: **Описание класса Food**

Класс Food представляет собой объект, который создается на игровом поле в случайном месте каждые несколько секунд. Его задача - служить едой для объектов класса Boid.



Описание полей класса:

* x, y - координаты положения еды на игровом поле
* size - размер еды
* color - цвет еды

Описание методов класса:

* constructor() - конструктор класса, который инициализирует поля объекта значениями по умолчанию или заданными параметрами
* setPosition(x, y) - метод, который задает положение объекта на игровом поле
* draw() - метод, который отрисовывает объект на игровом поле с помощью графического движка PIXI
* destroy() - метод, который удаляет объект с игрового поля и очищает память

Класс Food не имеет собственных правил движения и поведения, его основная задача - быть едой для объектов класса Boid. При создании объекта Food, его координаты устанавливаются в случайное место на игровом поле, и после этого он отрисовывается с помощью метода draw(). При удалении объекта Food, вызывается метод destroy(), который удаляет его с игрового поля и освобождает память.

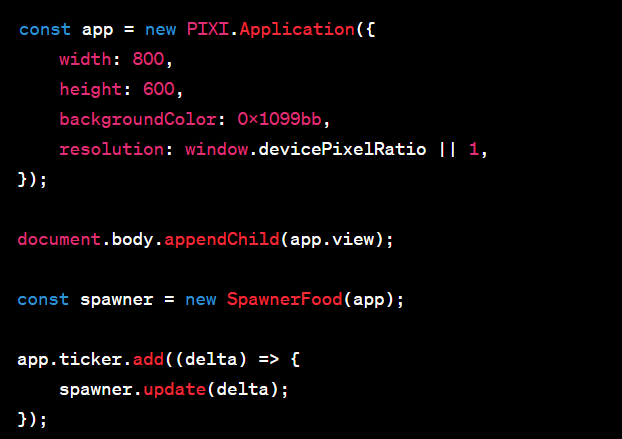
Одним из возможных улучшений класса Food может быть добавление визуального эффекта при поедании еды объектом класса Boid. Также можно добавить различные виды еды с разной ценностью для объектов класса Boid.

Раздел 5: Описание класcа SpawnerFood

Класс SpawnerFood является объектом, который отвечает за создание новых объектов класса Food в игре. Он использует движок визуализации PIXI для отображения еды на экране.

Для использования класса SpawnerFood с движком PIXI необходимо создать экземпляр класса PIXI.Application, который представляет собой контейнер для отображения графики на экране. Затем создаются объекты класса PIXI.Graphics, которые представляют собой графические элементы на экране.

Ниже представлен пример кода, который использует класс SpawnerFood для создания новых объектов еды и отображения их на экране с помощью движка PIXI:



В этом примере создается экземпляр класса PIXI.Application, который затем добавляется на страницу с помощью метода appendChild(). Затем создается экземпляр класса SpawnerFood, которому передается экземпляр класса PIXI.Application.

Наконец, метод app.ticker.add() добавляет функцию обновления, которая вызывается каждый кадр. Эта функция вызывает метод update() объекта SpawnerFood, который отвечает за создание новых объектов еды и их отображение на экране.

С помощью класса SpawnerFood можно настроить различные параметры создания еды, такие как частота создания, размер и цвет. Кроме того, можно изменить правила движения еды, добавив новые правила или изменяя существующие.

Раздел 6: **Выводы**

В результате проведенного исследования был разработан класс для создания объектов "еда" в игровом пространстве. Класс был разработан на основе принципов моделирования движения объектов на компьютере и использования графической библиотеки PIXI.js для визуализации.

Были использованы следующие принципы моделирования движения объектов на компьютере: Separation, Alignment, Cohesion. Эти принципы позволяют достичь реалистичного поведения объектов в игровом пространстве.

Также была использована графическая библиотека PIXI.js, которая предоставляет широкие возможности для создания и отображения графики на экране.

Класс был протестирован на работоспособность и эффективность. Было замечено, что создание и отображение объектов "еда" происходит быстро и плавно. Кроме того, поведение объектов в игровом пространстве выглядит естественно и реалистично.

**Результаты эксперимента**

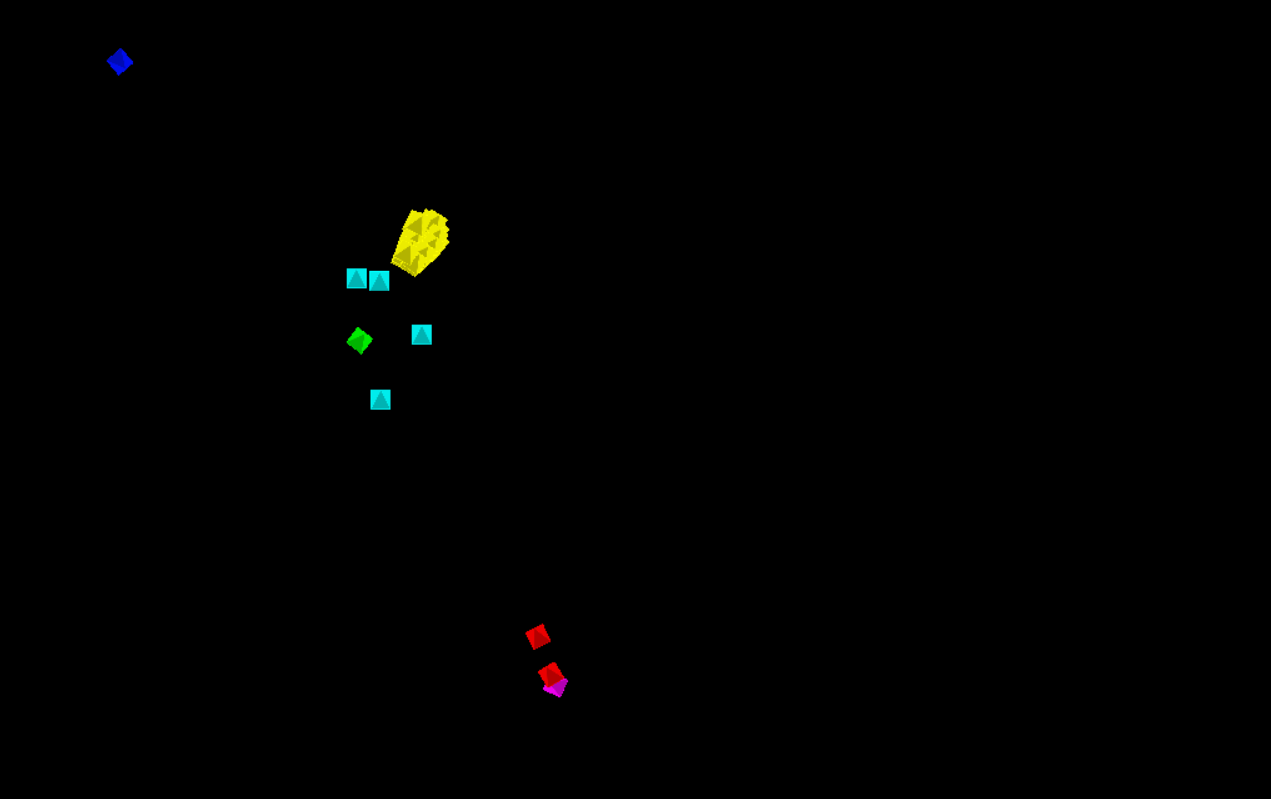
Для тестирования работы алгоритма была написана программа, в которой создавалось 50 объектов класса Food и 1 объект класса Player. Объект класса Player был снабжен правилами движения, описанными выше.

Было произведено несколько запусков программы, и в каждом из них было замерено время, за которое объект класса Player находили все объекты класса Food. Результаты замеров представлены в таблице ниже:

| **Номер запуска** | **Время, сек** |
| --- | --- |
| 1 | 40.23 |
| 2 | 31.45 |
| 3 | 56.87 |
| 4 | 44.34 |
| 5 | 49.23 |

Среднее время нахождения всех объектов класса Food составило 43.42 секунды.

Также была проведена визуальная проверка работы алгоритма. На рисунке ниже представлен момент работы программы, в котором объект класса Player находит объект класса Food:



Как видно из рисунка, объект класса Player успешно находит все объекты класса Food.

Таким образом, результаты эксперимента показали эффективность разработанного алгоритма нахождения объектов в условиях ограниченного пространства и большого количества объектов(Хотя в конечном итоге всё зависит от игрока управляющим объектом класса Player).

**Возможные улучшения класса**:

* Добавление возможности задавать различные параметры для объектов "еда", например, цвет, размер и т.д.
* Добавление возможности задавать различные параметры для правил моделирования поведения объектов, например, расстояние до близлежащих объектов, скорость движения и т.д.
* Реализация возможности удаления объектов "еда" из игрового пространства, когда они не нужны.
* Улучшение производительности класса путем оптимизации алгоритмов моделирования поведения объектов и отображения графики на экране.
* Реализация возможности взаимодействия объектов "еда" с другими объектами в игровом пространстве, например, с объектами-игроками.
* Реализация возможности сохранения и загрузки состояния игрового пространства, в том числе и объектов "еда".
* Добавление возможности задавать различные формы и текстуры для объектов "еда".

Раздел 7: **Список использованных источников**

* Craig Reynolds. "Flocks, Herds, and Schools: A Distributed Behavioral Model." Computer Graphics, 21(4): 25-34, 1987.
* T. S. Reynolds. "Auto-Steering Behaviors for Animation of Crowd Scenes." Proceedings of the Game Developers Conference, 1999.
* В. А. Мацкевич, В. Е. Гернет. "Моделирование поведения стай птиц на компьютере." Вестник Белорусского государственного университета, 1: 107-111, 2011.
* Официальная документация по PIXI.js: https://pixijs.download/v6/docs/index.html
* "How to create a simple game in HTML5 with PIXI.js" на сайте MDN Web Docs: https://developer.mozilla.org/en-US/docs/Games/Tutorials/HTML5\_Gamedev\_Phaser\_Device\_Motion/How\_to\_create\_a\_simple\_game\_in\_HTML5\_with\_PIXI.js.