МИНИСТЕРСТВО ОБРАЗОВАНИЯ НОВГОРОДСКОЙ ОБЛАСТИ

Областное государственное автономное профессиональное образовательное учреждение «Боровичский техникум строительной индустрии и экономики»

Специальность: 09.02.07 «Информационные системы и программирование»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА ПО**

**КУРСОВОМУ ПРОЕКТУ**

**Разработка программы Worm**

# КП 01 МДК.01.03. 02 2021 ПЗ

Разработал: студент группы П-31 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.К.Пустовойт

Руководитель курсового проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Ю.А. Архипова

Боровичи

2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ НОВГОРОДСКОЙ ОБЛАСТИ

Областное государственное автономное профессиональное образовательное учреждение «Боровичский техникум строительной индустрии и экономики»

«Утверждаю»

Зам. директора по учебной работе (зав. отделением) М.Ю. Кутузова

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на курсовое проектирование**

**МДК 01.03 Разработка мобильных приложений**

**«Разработка программы Worm»**

Студенту группы П-31 Пустовойт Сергей Константинович

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Введение

1. Общая часть
   1. Постановка задачи
      1. Назначение разработки
      2. Функциональные требования к программе
   2. Обзор существующих решений
   3. Технические и программные средства
      1. Требования к составу и параметрам технических средств
      2. Требования к программной и информационной совместимости
2. Специальная часть
   1. Описание алгоритмов
   2. Технология разработки программного изделия
   3. Результаты работы программы и их оценка
3. Технологическая часть
   1. Технология нисходящего проектирования программных изделий

Заключение

Приложение А. Текст программы

Приложение Б. Видовые экраны работы программы

Приложение В. Алгоритм решения задачи

Дата выдачи «30» октября 2021 г.

Срок окончания «11» декабря 2021 г.

Преподаватель-руководитель курсового проектирования Ю.А. Архипова

Председатель цикловой комиссии С. В. Андреева

# Аннотация

Данный курсовой проект разрабатывался на платформe Swift9 . Для работы с данным проектом необходимо было изучить тонкости языка программирования и выбранной платформы для разработки.

Работа проводилась с реальными источниками и информацией для более реальной и удобной работы программы. Все прописанные в коде функции и операции являются необходимыми для корректной работы программы.

Количество рисунков в данном курсовом проекте составляет два. Они отражают видовые окна программы и показывают, как выглядит интерфейс. Присутствуют рисунки, которые показывают код программы, вызов определённых обработчиков событий, Все рисунки находятся в приложении.

Объём работы составляет 30 страницы.

Содержание

[Аннотация 4](#_Toc74174748)

[Содержание 5](#_Toc74174749)

[Введение 3](#_Toc74174750)

[1. Общая часть 5](#_Toc74174751)

[2. Специальная часть 11](#_Toc74174752)

[3. Технологическая часть 14](#_Toc74174753)

[3.1 Технологии визуального проектирования приложений 14](#_Toc74174754)

[4.1Эргономическая безопасность дисплеев 19](#_Toc74174757)

[4.2Гигиеническая оценка рабочих мест по условиям труда 21](#_Toc74174758)

[Заключение 24](#_Toc74174759)

[Использованная литература 26](#_Toc74174760)

[Приложение А. Текст программы 27](#_Toc74174761)

[Приложение Б. Видовые экраны работы программы 36](#_Toc74174762)

[Приложение В. Алгоритм решения задачи 39](#_Toc74174763)

# Введение

Данная программа является классической в своём жанре подобных.

## Программа "Worm" осуществляет вывод на экран червяка который поебает красные квадраты. Назначение программы – развлечение.

Цель – разработать видеоигру Worm в классическом исполнении.

Задачи:

1. Изучить основы

2. Ознакомится с оформлением мобильных приложений

3. Ознакомится и углубиться в язык программирования Swift

4. Изучить предметную область

Создание видео игры Worm.

Объектом курсовой работы будет являться создание программы с воздействием пользователя на события происходящие в режиме реального времени, именование программы: Worm. Под объектом курсовой понимается создание программы и ознакомление с языком swift.

В данном курсовом проекте предметом рассмотрения и изучения будут являться механики языка swift.

Главная задача всего проекта является: осуществить полноценную работоспособность программы для возможности взаимодействия с ней пользователя.

Практическая значимость. Данная программа предоставляет возможность развлечения и расслабления.

Разрабатываемая программа предназначена для использования в целях развлечься.

Программный продукт должен использоваться исключительно на айфонах, в данном случае только с программным обеспечением IOS.

Интерфейс интуитивно понятен и имеет приятное оформление, теплые глазу цвета и цветовую гамму в целом. При включении пользователь может видеть экран загрузки, где представлен логотип и название «Worm»

# Общая часть

* 1. **Постановка задачи**
     1. **Назначение разработки**

Требуется создать мобильное приложение для развлечения и расслабления, которое позволит успокоиться.

«Worm»­ ­ – программный продукт разработанный на платформе Xcode и языке программирования Swift. Курсовая работа с программой «Worm» предназначена для расслабления. Данный курсовой проект предназначен для создания программного продукта, который позволит расслабиться и развлечься после тяжелого дня «Worm», которая предназначена для любого возраста.

.

«Worm» решает все введенные данные, анализирует какой график требуется и строит его, тем самым помогая или делая работу за пользователя.

* + 1. **Функциональные требования к программе**

«Worm» – программа видео-игра . Это приложение предназначено для пользования в личных целях.

Данный курсовой проект включает в себя создание программы «Worm», которая будет состоять из нескольких модулей:

Экран загрузки

Экран игры

Таблица 1 – Варианты использования пользователем

|  |  |
| --- | --- |
| Раздел 1. Описание варианта использования | |
| Наименование варианта использования | Подсчёт |
| Актеры | Пользователь |
| Краткое описание | Пользователь управляет движение червя |
| Цель варианта использования | Вычисление входящих значений |
| Тип варианта использования | Базовый |
| Ссылки на другие варианты использования | Нет |
| Раздел 2. Последовательность действий актеров | |
| Действия актера | Отклик системы |
| 1. Пользователь вносит данные | 1.1 Записывание информации |
| 2. Получение ответа | 2.1 Демонстрация вычислений программой |
| Раздел 3. Последовательность действий актеров при возникновении исключительных ситуаций | |
| Действия актеров | Отклик системы |

Таблица 2 – Варианты использования пользователем

|  |  |
| --- | --- |
| Раздел 1. Описание варианта использования | |
| Наименование варианта использования | Построение графика функции |
| Актеры | Пользователь |
| Краткое описание | Пользователь управляет движение червя |
| Цель варианта использования | Вычисление входящих значений |
| Тип варианта использования | Базовый |
| Ссылки на другие варианты использования | Вычисления |
| Раздел 2. Последовательность действий актеров | |
| Действия актера | Отклик системы |
| Раздел 3. Последовательность действий актеров при возникновении исключительных ситуаций | |
| Действия актеров | Отклик системы |

* 1. **Технические и программные средства**
     1. **Требования к составу и параметрам технических средств**

Для полноценной работы приложения необходим смартфон с операционной системой IOS.

* + 1. **Требования к программной и информационной совместимости**

«Worm» работает только с платформой Xcode. Пользователь должен иметь смартфон айфон 12 и модели для работы с программой.

# Специальная часть

* 1. **Организация данных и диалога**

Организация данных и диалог с пользователем осуществлён таким образом,

для того чтобы ввести данные пользователь должен вбить их при помощи специальной клавиатуры, но вводимые данные не хранятся в программе.

Диалог происходит посредством подсчёта данных, вывода ответа и вывода графика.

* 1. **Технология разработки программного изделия**

При разработке данного приложения использовались такие элементы как, экран загрузки, экран игры.

Эмулятором выбран айфон12, на данном смартфоне отображаются все основные компоненты приложения. Данная программа не требует работы с сетью Интернет или с библиотеками, ведь все функции являются основными и их можно прописать, зная формулы и математические функции.

Технология разработки заключалась в изначальном создании макета приложения в AdobeXD, переносе графического интерфейса, составлении переходов между видовыми окнами и дальнейшего кодирования программы.

* 1. **Результаты работы программы и их оценка**

Работа программы начинается с её запуска и демонстрации экрана загрузки. На нём показан логотип приложения и его название.

Далее перед пользователем открывается главное окно приложения, где пользователь начинает играть. Данное окно оформлено интуитивно просто, окрашен в приятные пользователю цвета.

# Технологическая часть

# 3.1 Технологии визуального проектирования приложений

Визуальное программирование (от лат. visualis – зрительный) – это технология программирования, предусматривающая создание приложений с помощью наглядных средств.

К визуальному программированию можно отнести также Rapid Application Development (RAD) – быструю разработку программ. RAD – технология программирования, обеспечивающая ускоренную разработку и модификацию приложений за счет использования объектно-ориентированного и визуального программирования.

Средствами визуального программирования обычно решают задачи построения пользовательского интерфейса и упрощения разработки приложения путем замены метода "написания программы" на метод конструирования.

Визуальное программирование, бесспорно, обладает достоинством наглядного представления информации и гораздо лучше соответствует природе человеческого восприятия, чем методы традиционного, текстового программирования. Однако практически все визуальные средства нуждаются в дополнении функциями, которые не могут быть представлены в виде графических конструкций и требуют текстового выражения. Визуальные средства дополняются специальными программами – "скриптами", написанными на различных языках программирования.

Концепция визуального программирования реализована во многих современных средах разработки программных систем. Все ведущие фирмы, создающие средства для программирования и конструирования имеют системы, поддерживающие технологию визуального программирования.

Начальные шаги технологии визуального программирования определяются оболочкой самой среды визуального программирования. Сначала создаются экранные формы простейшей буксировкой мыши. В инспекторе объектов производится настройка их свойств путем заполнения отдельных полей. На главную форму помимо визуальных компонент наносятся не визуальные компоненты. Формы объединяются в единый проект. Далее в соответствии со сценарием диалога программируются методы события основной и подчиненных форм. Программы «пустых» методов событий появляются в окне редактора после нажатия соответствующих клавиш или действий мыши. «Пустые» методы дополняются определенными операторами активации и дезактивации форм. По окончании начальных шагов получается работающий «скелет» программы с источниками данных из файловых баз данных и со сгенерированными формами документов, выводимых на печать. Исследователь (Browser) обеспечивает визуализацию схемы иерархии классов полученного «скелета» программы. Другими словами, технический проект реализованной части программы формируется автоматически.

Дальнейшая разработка программы ведется по технологии объектно-ориентированного программирования. Можно часть программы реализовать по технологии структурного программирования. Некоторые недостающие визуальные и невизуальные компоненты получаются модификацией исходных текстов наиболее близких прототипов имеющихся компонент. Рекомендуется новые компоненты помещать в палитру компонент. Это облегчит их повторное использование в данной или последующих разработках. Код, относящийся только к данной разработке, набирается по тексту программы.

Выводы:

• Визуальное программирование во многом автоматизирует труд программиста по написанию программ.

• Визуальное программирование — одна из самых популярных парадигм программирования на данный момент. Оно базируется на технологии ООП.

• Среда визуального программирования поддерживает работу браузеров (Browser), при помощи которых можно автоматически получить документацию по структуре программы.

• Основным элементом в средствах визуального программирования является компонент. Компоненты бывают визуальными и не визуальными.

• Технология визуального программирования состоит в следующем: создание экранных форм, нанесение визуальных и не визуальных компонент, программирование событий и методов оконных форм.

# 4.1Эргономическая безопасность дисплеев

Эргономическая безопасность характеризуется такими условиям работы, когда ее удобно выполнять с максимальным эффектом. Это достигается правильным размещением рабочего стола и применением технических средств.

Общие эргономические требования и требования безопасности. Важным условием безопасности человека перед экраном является правильный выбор визуальных параметров дисплея и светотехнических условий рабочего места. В ГОСТ Р 50948-96 и ГОСТ Р 50949-96 и в Санитарных правилах и нормах (СанПиН) установлены требования к двум группам визуальных параметров:

1) Яркость, освещенность, угловой размер знака и угол наблюдения.

2) Неравномерность яркости, блики, мелькания, расстояние между

знаками, словами, строками, геометрические и нелинейные искажения, дрожание изображения и т.д.

Вторая группа требований обеспечения эргономической безопасности - нормы на излучения.

Визуальные эргономические параметры ПК являются параметрами безопасности, и их неправильный выбор приводит к ухудшению здоровья пользователей. Опасности, которые вас подстерегают, могут быть связаны с обострением хронических заболеваний глаз, проявлением наследственных предрасположенностей. Поэтому так важен режим работы с ПК, профилактические мероприятия, ну и, конечно, самое главное — эргономические параметры видеомониторов.

Одним из основных параметров является частота вертикальной или кадровой развертки (частота обновления), которая (что признается подавляющим числом международных и национальных стандартов) должна быть не менее 85 Гц, желательно в режиме максимального разрешения.

Особенно это важно при работе с графическими пакетами. В практической эргономике для определения усталостных характеристик человека-оператора применяется такой психофизиологический показатель, как критическая частота слияния мельканий (КЧСМ). Так вот КЧСМ зависит от яркости изображения, спектра излучения, местоположения изображения на сетчатке глаза, размеров наблюдаемого объекта, от возраста оператора и ряда других факторов, в том числе от времени работы человека с информационной моделью, вызывающей усталость. Но в любом случае она не может превысить 30-35 Гц в центральной области зрения. Этими цифрами, как вы уже догадались, определяется нижняя граница допустимой частоты вертикальной развертки монитора. Особенно высокой чувствительностью к изменению яркости изображения обладают окраинные области сетчатки ПК.

В настоящее время все мониторы должны соответствовать стандарту MPRII, ограничивающему излучение мониторов в диапазоне крайне низких частот (некоторые основные параметры, определенные этим и другими стандартами.

Требования эргономики и технической эстетики к рабочему месту программиста:

Для обеспечения требований эргономики и технической эстетики конструкция рабочего места, расположение и конструкция органов управления должны соответствовать анатомическим и психофизиологическим характеристикам человека. Вместе с этим все оборудование, приборы и инструменты не должны вызывать психологических раздражений.

Основными элементами рабочего места оператора являются: рабочий стол, рабочий стул (кресло), дисплей, клавиатура.

# 

# 4.2Гигиеническая оценка рабочих мест по условиям труда

Гигиеническая оценка условий труда заключается в определении фактических значений опасных и вредных производственных факторов на рабочих местах.

При аттестации рабочего места по условиям труда оценке подлежат все имеющиеся на рабочем месте опасные и вредные производственные факторы (физические, химические, биологические), тяжесть и напряженность труда.

Прежде всего рабочие места проверяются по гигиеническим критериям. Они показывают, насколько условия труда отличаются от действующих гигиенических нормативов.

При проверке гигиенических условий следует опираться на:

- Руководство по гигиенической оценке факторов рабочей среды и трудового процесса № Р 2.2.2006-05;

- Межотраслевые правила по охране труда в общественном питании ПОТ РМ-011-2000;

- Межотраслевые правила по охране труда в розничной торговле ПОТ РМ-014-2000.

При аттестации рабочих мест по условиям труда оценке подлежат все имеющиеся на рабочем месте вредные и (или) опасные производственные факторы.

Вредными производственным факторам могут быть:

- физические – такие как температура, влажность, скорость движения воздуха, тепловое излучение, производственный шум, освещение и т.п.;

- химические – химические вещества, смеси, получаемые химическим синтезом и (или) для контроля которых используют методы химического анализа;

- биологические факторы- микроорганизмы-продуценты, живые клетки и споры, содержащиеся в бактериальных препаратах, патогенные микроорганизмы - возбудители инфекционных заболеваний;

- тяжесть – преимущественно нагрузка на опорно–двигательный аппарат и функциональные системы организма (сердечно-сосудистую, дыхательную и др.), обеспечивающие его деятельность;

- напряженность;

- нагрузка преимущественно на центральную нервную систему, органы чувств, эмоциональную сферу работника, например, интеллектуальные, сенсорные, эмоциональные нагрузки, степень монотонности нагрузок, режим работы.

Опасные факторы рабочей среды

- факторы среды и трудового процесса, которые могут стать причиной острого заболевания или внезапного резкого ухудшения здоровья, смерти. В зависимости от количественной характеристики и продолжительности действия отдельные вредные факторы рабочей среды могут стать опасными.

Исходя из степени отклонения фактических уровней факторов рабочей среды и трудового процесса от гигиенических нормативов условия труда по степени вредности и опасности условно подразделяются на 4 класса: оптимальные, допустимые, вредные и опасные.

Оптимальные условия труда (1 класс) – условия, при которых сохраняется здоровье работника и создаются предпосылки для поддержания высокого уровня работоспособности. Оптимальные нормативы факторов рабочей среды установлены для микроклиматических параметров и факторов трудовой нагрузки. Для других факторов за оптимальные условно принимают такие условия труда, при которых вредные факторы отсутствуют либо не превышают уровни, принятые в качестве безопасных для населения.

Допустимые условия труда (2 класс) характеризуются такими уровнями факторов среды и трудового процесса, которые не превышают установленных гигиенических нормативов для рабочих мест, а возможные изменения функционального состояния организма восстанавливаются во время регламентированного отдыха или к началу следующей смены и не оказывают неблагоприятного действия в ближайшем и отдаленном периоде на состояние здоровья работников и их потомство. Допустимые условия труда условно относят к безопасным.

Вредные условия труда (3 класс) характеризуются наличием вредных факторов, уровни которых превышают гигиенические нормативы и оказывают неблагоприятное действие на организм работника и/или его потомство

# Заключение

Основной целью данного курсового проекта является разработка программного обеспечения для построения графиков математических функций. При разработке необходимо было узнать предметную область работы приложения, основную аудиторию и понять главные задачи работы.

Сначала нужно было определить цель и функции курсового проекта и максимально их осуществить. В процессе разработки функции программы менялись несколько раз, пока не выяснились основные функции, а остальные нужно было удалить, так как они были бесполезны и трудноосуществимы.

Далее следовало составление таблиц и диаграмм. Разработанное приложение не имеет хранимых данных, большое количество пользователей с разными функциями и задачами, поэтому были составлены: диаграмма прецедентов и таблица вариантов использования.

На следующем этапе надо было решить алгоритмы решения задач, так как программа имеет работу с математикой, то надо было составить блок-схемы для решения простых вычислений и построения. Все данные в блок-схемах имеют общий вид для всех операций и функций, ведь они работают по одному алгоритму.

В данной курсовой работе рассмотрены другие программы, которые являются аналогами приложения. Разработанная программа является уникальной, из-за её доступности на Android и абсолютная бесплатность, своя цветовая гамма и свой набор функций, расположение клавиш и полей ответа. Другие решения являются программами крупных компаний - производителей ПО и они доступны на всех платформах.

Неотъемлемой частью данного курсового проекта является описание функционала и технических требований, последнее включает умение работы с описанными характеристиками. Функционал расписан достаточно полно и ясно. Функциональные требования к пользователю являются одними из важных, ведь они включают навыки работы с девайсами, а также аккаунтов, при помощи которых можно установить данное ПО.

В технологической части проекта описана и раскрыта тема визуального программирования, а точнее технология визуального проектирования приложений и технология проектирования баз данных. В данном разделе раскрыты данные темы и также связаны с практикой, так как это основа разработки и проектирования программы. Описывается теоретическая часть основ визуального проектирования и проектирования баз данных.

Пункт «Результаты» включает в себя итоги работы всех функций и их оценку. Здесь подробно расписаны действия пользователя и его контакт с программой. Данный пункт связан с технологией разработки и организации данных и диалога с пользователем, ведь идёт оценка разработанного приложения и диалога программы с пользователем.

Курсовой проект позволил посмотреть на работу других схожих ПО и понять их работу, а после разработать своё. Цели и задачи, поставленные вначале выполнены на максимум и в дальнейшем имеет место для модернизации самой программы, возможным добавлением новых функций.

# Использованная литература

1. Интернет-учебник по программированию под Xcode [Электронный ресурс] Режим доступа: https://2compa.ru/programmirovanie/xcode-and-swift/interfeis-ide-xcode-i-shablony/.

2. Документация по языку программирования Swift [Электронный ресурс] Режим доступа: http:// www.apple.com/ru/swift/

3. Сайт вопросов и ответов для программистов [Электронный ресурс] Режим доступа: https://ru.stackoverflow.com

4.Оценка условий труда по гигиеническим критериям [Электронный ресурс] Режим доступа: https://studfile.net/preview/6381613/page:50/

5.Требования безопасности к излучению от дисплея [Электронный ресурс] Режим доступа: https://studbooks.net/2429196/informatika/trebovaniya\_bezopasnosti\_izlucheniyu\_displeya

6.Визуальное программирование [Электронный ресурс] Режим доступа: https://studopedia.ru/3\_115492\_vizualnoe-programmirovanie.html

7.Технология визуального программирования [Электронный ресурс] Режим доступа: https://it.wikireading.ru/58399

# Приложение А. Код программы

**GameScene.swift**

//  
// GameScene.swift  
// Worm  
//  
//  
  
import SpriteKit  
import GameplayKit  
import Darwin  
  
struct PCategory {  
static var foodCategory:UInt32 = 0x1  
static var framesCategory: UInt32 = 0x1 « 1  
static var snakeCategory: UInt32 = 0x1 « 2  
}  
  
struct Point {  
var node:SKSpriteNode  
var x:Int  
var y:Int  
  
func setPhysics(\_ category: UInt32, \_ contactTest: UInt32, \_ isDynamic: Bool) {  
node.physicsBody = SKPhysicsBody(rectangleOf: CGSize(width: node.size.width - 1, height: node.size.height - 1))  
node.physicsBody?.affectedByGravity = false  
node.physicsBody?.categoryBitMask = category  
node.physicsBody?.contactTestBitMask = contactTest  
node.physicsBody?.collisionBitMask = 0  
node.physicsBody?.isDynamic = isDynamic  
}  
}  
  
class GameScene: SKScene, SKPhysicsContactDelegate {  
var food:Point!  
var snake: [Point] = []  
var frames: [Point] = []  
private static let POINT\_SIZE = 10  
  
func didBegin(\_ contact: SKPhysicsContact){  
let foodAndSnake = PCategory.foodCategory | PCategory.snakeCategory  
let framesAndSnake = PCategory.snakeCategory | PCategory.framesCategory  
switch contact.bodyA.categoryBitMask | contact.bodyB.categoryBitMask {  
case foodAndSnake:  
spawnFood()  
grouSnake()  
case framesAndSnake:  
layoutScene()  
case PCategory.snakeCategory:  
layoutScene()  
default:  
print("Contact O\_o")  
}  
}  
  
func grouSnake(){  
let point = createYellowPoint(x: 1000, y: 1000)  
addChild(point.node)  
snake.append(point)  
}  
  
override func touchesBegan(\_ touches: Set<UITouch>, with event: UIEvent?) {  
for t in touches {  
let pos = t.location(in: self)  
touchDown(pos)  
}  
}  
  
func touchDown( \_ pos: CGPoint){  
let p = ( Double(pos.x - self.frame.midX), Double(pos.y - frame.midY))  
switch p {  
case let (x,y) where y > 0 && y > abs(x) :  
dX = 0  
dY = GameScene.POINT\_SIZE  
case let (x,y) where y < 0 && abs(y) > abs(x) :  
dX = 0  
dY = -GameScene.POINT\_SIZE  
case let (x,y) where x > 0 && x > abs(y) :  
dX = GameScene.POINT\_SIZE  
dY = 0  
case let (x,y) where x < 0 && abs(x) > abs(y) :  
dX = -GameScene.POINT\_SIZE  
dY = 0  
default:  
print("Center O\_o")  
}  
}  
  
override func didMove(to view: SKView) {  
self.physicsWorld.contactDelegate = self  
layoutScene()  
}  
  
var dX = GameScene.POINT\_SIZE  
var dY = 0  
  
func moveSnake(){  
var x = 0.0  
var y = 0.0  
var head = true  
for p in snake {  
let a = head ?  
SKAction.move(by: CGVector(dx:dX,dy:dY), duration: 0) :  
SKAction.move(to: CGPoint(x:x,y:y), duration: 0)  
x = Double(p.node.position.x)  
y = Double(p.node.position.y)  
[p.node.run](https://vk.com/away.php?to=http%3A%2F%2Fp.node.run&cc_key=)(a)  
head = false  
}  
}  
  
var timeDelta:TimeInterval = 0  
var oldTime:TimeInterval = 0  
  
override func update(\_ currentTime: TimeInterval) {  
super.update(currentTime)  
timeDelta = timeDelta + (currentTime - oldTime)  
oldTime = currentTime  
if timeDelta > 0.2 {  
moveSnake()  
timeDelta = 0  
}  
}  
  
func layoutScene(){  
backgroundColor = UIColor(red: 0, green:0, blue:0, alpha:1.0)  
dX = GameScene.POINT\_SIZE  
dY = 0  
spawnFood()  
spawnFrames()  
spawnSnake()  
}  
  
func spawnFood(){  
if food != nil{  
food.node.removeFromParent()  
food = nil  
}  
let node = SKSpriteNode(  
color: UIColor(red:1.0, green:0.0, blue: 0.0, alpha: 1.0),  
size: CGSize(width: GameScene.POINT\_SIZE, height: GameScene.POINT\_SIZE)  
)  
let x = arc4random\_uniform(UInt32((frame.width - CGFloat( 3\*GameScene.POINT\_SIZE))))  
let y = arc4random\_uniform(UInt32((frame.height - CGFloat( 3\*GameScene.POINT\_SIZE))))  
node.position = CGPoint(x: Int( x), y:Int( y))  
addChild(node)  
food = Point(node: node, x: Int(x), y: Int(y))  
food.setPhysics(PCategory.foodCategory, PCategory.snakeCategory, false)  
}  
  
func spawnFrames() {  
for f in frames{  
f.node.removeFromParent()  
}  
frames = []  
let width = Int(frame.width)  
let height = Int( frame.height)  
createHorisontalFrame(y: 0, minX: 0, maxX: width)  
createHorisontalFrame(y: height, minX: 0, maxX: width)  
createVerticalFrame(x: 0, minY: 0, maxY: height)  
createVerticalFrame(x: width, minY: 0, maxY: height)  
}  
  
func spawnSnake(){  
for s in snake{  
s.node.removeFromParent()  
}  
snake = []  
createSnake(x: Int(frame.midX), y: Int(frame.midY))  
}  
  
func createBluePoint(x:Int, y:Int) ->

Point {  
let node = SKSpriteNode(  
color: UIColor(red:0.0, green:0.0, blue: 1.0, alpha: 1.0),  
size: CGSize(width: GameScene.POINT\_SIZE, height: GameScene.POINT\_SIZE)  
)  
node.position = CGPoint(x: x, y: y)  
let point = Point(node: node, x: x, y: y)  
point.setPhysics(PCategory.framesCategory, PCategory.snakeCategory, false)  
return point  
}  
  
func createYellowPoint(x:Int, y:Int) -> Point {  
let node = SKSpriteNode(  
color: UIColor(red:1.0, green:1.0, blue: 0.0, alpha: 1.0),  
size: CGSize(width: GameScene.POINT\_SIZE, height: GameScene.POINT\_SIZE))  
node.position = CGPoint(x: x, y: y)  
let point = Point(node: node, x: x, y: y)  
  
point.setPhysics(PCategory.snakeCategory, PCategory.foodCategory | PCategory.framesCategory | PCategory.snakeCategory, true)  
point.node.physicsBody?.usesPreciseCollisionDetection = true  
return point  
}  
  
func createHorisontalFrame(y:Int, minX:Int, maxX: Int){  
for i in stride(from: minX, to: maxX, by: GameScene.POINT\_SIZE){  
let point = createBluePoint(x: i, y: y)  
addChild(point.node)  
frames.append(point)  
}  
}  
  
func createVerticalFrame(x:Int, minY:Int, maxY: Int){  
for i in stride(from: minY, to: maxY, by: GameScene.POINT\_SIZE){  
let point = createBluePoint(x: x, y: i)  
addChild(point.node)  
frames.append(point)  
}  
}  
  
func createSnake(x:Int, y:Int){  
for i in 0...2 {  
let point = createYellowPoint(x: x - GameScene.POINT\_SIZE\*i, y: y)  
if i == 0 {  
point.node.color = UIColor(red:1.0, green: 0.0, blue: 0.5, alpha: 1.0)  
}  
addChild(point.node)  
snake.append(point)  
}  
}  
}

**GameViewController.swift**

//  
// GameViewController.swift  
// Worm  
//  
//  
  
import UIKit  
import SpriteKit  
import GameplayKit  
  
class GameViewController: UIViewController {  
  
override func viewDidLoad() {  
super.viewDidLoad()  
  
if let view = self.view as! SKView? {  
  
let scene = GameScene(size: view.bounds.size)  
  
scene.scaleMode = .aspectFill  
  
  
view.presentScene(scene)  
  
  
view.ignoresSiblingOrder = true  
  
view.showsFPS = true  
view.showsNodeCount = true  
}  
  
}  
  
override var shouldAutorotate: Bool {  
return true  
}  
  
override var supportedInterfaceOrientations: UIInterfaceOrientationMask {  
if UIDevice.current.userInterfaceIdiom == .phone {  
return .allButUpsideDown  
} else {  
return .all  
}  
}  
  
override func didReceiveMemoryWarning() {  
super.didReceiveMemoryWarning()  
  
}  
  
override var prefersStatusBarHidden: Bool {  
return true  
}  
}

**AppDelegate.swift**

//  
// GameViewController.swift  
// Worm  
//  
//  
  
import UIKit  
import SpriteKit  
import GameplayKit  
  
class GameViewController: UIViewController {  
  
override func viewDidLoad() {  
super.viewDidLoad()  
  
if let view = self.view as! SKView? {  
  
let scene = GameScene(size: view.bounds.size)  
  
scene.scaleMode = .aspectFill  
  
  
view.presentScene(scene)  
  
  
view.ignoresSiblingOrder = true  
  
view.showsFPS = true  
view.showsNodeCount = true  
}  
  
}  
  
override var shouldAutorotate: Bool {  
return true  
}  
  
override var supportedInterfaceOrientations: UIInterfaceOrientationMask {  
if UIDevice.current.userInterfaceIdiom == .phone {  
return .allButUpsideDown  
} else {  
return .all  
}  
}  
  
override func didReceiveMemoryWarning() {  
super.didReceiveMemoryWarning()  
  
}  
  
override var prefersStatusBarHidden: Bool {  
return true  
}  
}

//  
// AppDelegate.swift  
// Worm  
//  
// Created by Vanquisher Winbringer on 03.07.2018.  
// Copyright © 2018 Vanquisher Winbringer. All rights reserved.  
//  
  
import UIKit  
  
[@UIApplicationMain](https://vk.com/id136052326)  
class AppDelegate: UIResponder, UIApplicationDelegate {  
  
var window: UIWindow?  
  
func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
  
return true  
}  
  
func applicationWillResignActive(\_ application: UIApplication) {  
}  
  
func applicationDidEnterBackground(\_ application: UIApplication) {  
  
}  
  
func applicationWillEnterForeground(\_ application: UIApplication) {  
  
}  
  
func applicationDidBecomeActive(\_ application: UIApplication) {  
  
}  
  
func applicationWillTerminate(\_ application: UIApplication) {  
  
}  
  
}

# Приложение Б. Видовые экраны работы программы

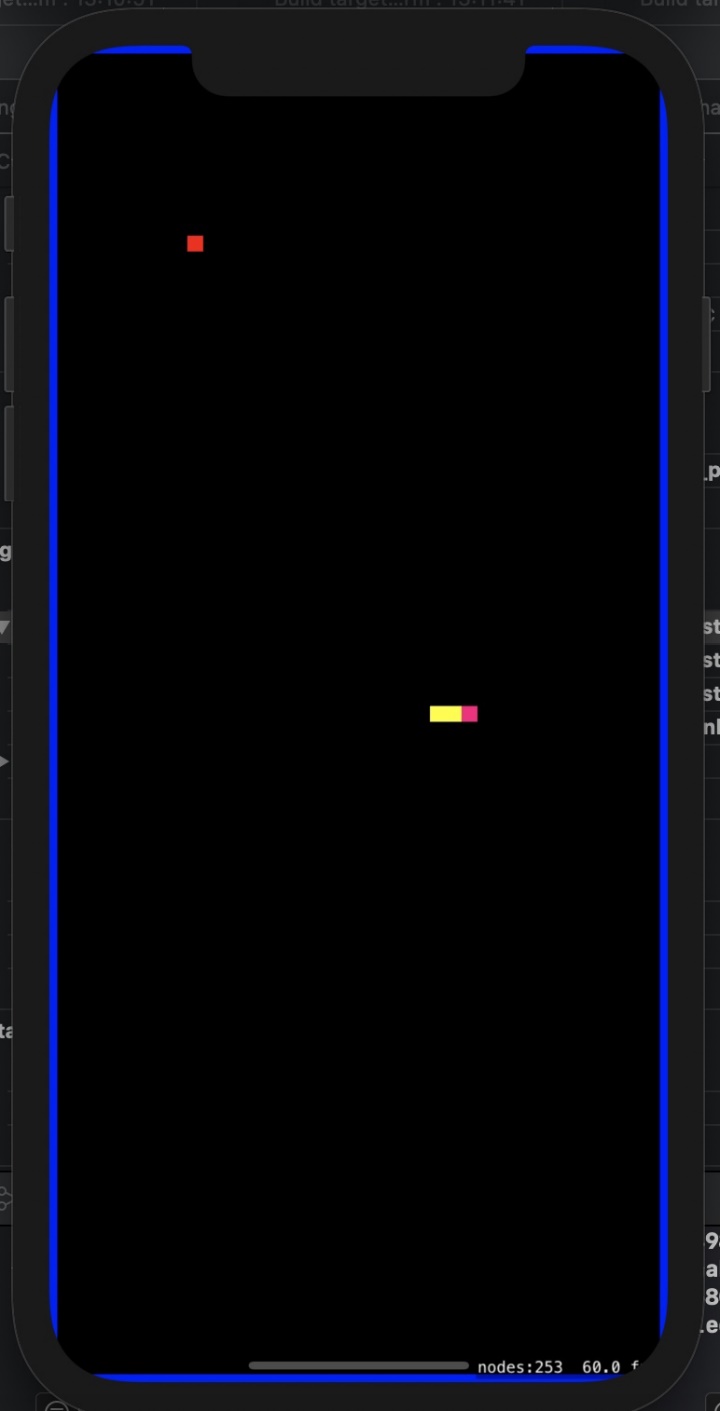


рис.1 Экран загрузки



рис.2 Экран игры

# Приложение В. Алгоритм игры

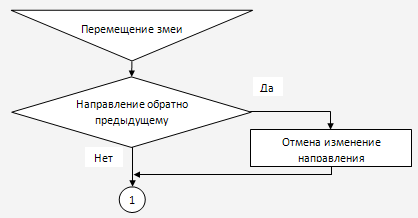


рис.1 Общий алгоритм игры