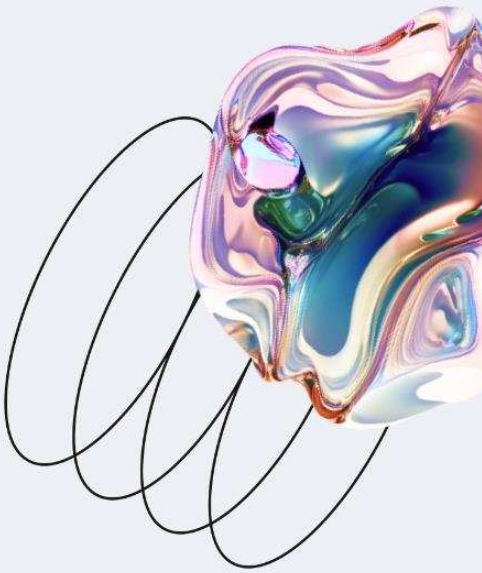
GeekBrains



Итоговая работа по курсу «Программирование на языке С. Продвинутый уровень»

Минчук Кирилл Евгеньевич

Описание проекта

Проект содержит описание и исходный код системы управления сельскохозяйственными дронами, далее кратко АСУ "Сельхоздрон" или просто АСУ.

Проект разработан в рамках курсовой работы по предмету "Программирование на языке С. Продвинутый уровень", в соответствии с Техническим заданием.

АСУ "Сельхоздрон" - это компьютерная симуляция сбора тыкв дронами. Дронами с 4 полей собираются спелые тыквы и кладутся в вереницу, прикрепленных к ним, тележек. Одна собранная тыква увеличивает вереницу на одну тележку.

Визуальные объекты симуляции расположены в 2-D пространстве и разделены на следующие группы:

- Дроны
- Тыквы
- Поля
- Текстовая информация

Дронами можно управлять как в ручном режиме, так и в автоматическом. В процессе сбора спелых тыкв, созревают новые. После сбора всего урожая выводится информация о количестве тыкв, собранных каждым дроном.

Системные требования и библиотеки

АСУ работает в консольном режиме, использую режим псевдографики.

АСУ написана на языке СИ с использованием одной внешней и 4 стандартных библиотек:

- ncurses
- stdio
- locale
- math

Проверка работоспособности проводилась на IBM PC-совместимом компьютере под управлением операционной системы XUBUNTU 20.04.6 LTS с установленным пакетом libncursesw5.

В онлайн-компиляторе запуск АСУ не представляется возможным из-за модульности или разбивки программы на отдельные файлы.

АСУ основано на консольной игре змейка. Но, имеет сильно переработанный и дополненный алгоритм работы. По сравнению с 750 строками змейки, АСУ имеет почти в два раза больше строк кода - 1340.

Отличия от змейки

Несмотря на то, что проект змейки был реализован ранее, несколько требований и пожеланий практически в корне изменили основную логику.

И одно из таких пожеланий было, наличие дорожек между грядками или как их назвали в ТЗ "границы тыквенного поля".

В АСУ реализован модуль maps.c, содержащий функции помогающие рисовать карту полей. Функции модуля позволяют генерировать достаточно сложные объекты и геометрические фигуры. Но, для сокращения времени реализации проекта были выбраны достаточно простые 4 сельхозполя в форме прямоугольников.

Чтобы сельхозполя и другие фигуры приобрели физические свойства препятствий, был реализован стоп-лист, который представляет из себя двухмерный массив, содержащий координаты всех необходимых объектов. Данный стоп-лист меняется динамически. Например, такие объекты как еда имеют четыре вида:

1. зеленый вьющейся побег
2. желтый цветок
3. зрелая красная тыква
4. темное место отрыва от стебля

На первые три объекта нельзя наступать, т.к это их погубит. Можно ходить по свободным ячейкам, местам срыва и в ячейку тыквы для ее сбора. Остальные места нужно бережно обходить. По мере созревания цветок превращается в тыкву, а извилистый побег зацветает.

В отличие от змейки тут нет уровней и ускорений и задержка кадра составляет 100мс в соответствии с ТЗ.

На верху экрана, в процессе сбора тыкв, отображается текстовая информация: Название дрона, его логотип и сколько тыкв он собрал.

В соответствии с заданием реализован режим предупреждения о столкновении isCrush(), внизу экрана, в момент коллизии вместе с именем дрона отобразится знак кирпича ⛔.

🕛 Слева внизу экрана отображается секундомер пройденного с начала игры времени.

Самая главная особенность проекта - дроны реализованы с помощью юникод эмодзи.

Возникшие проблемы

Отображение эмодзи

Стандарт кодирования символов unicode или UTF, включает не только набор букв и символов для большинства языков, а также иероглифы, пиктограммы и эмодзи.

Эмодзи или в простонародье "смайлики" это специальные символы, в большинстве случаев, представляющие собой изображения лиц и их эмоции. К эмодзи относят не только лица, но и предметы, фигурки людей или животных и др.

Unicode подразделяется в зависимости от размера символа на следующие группы шрифтов:

- UTF-8
- UTF-16
- UTF-32

В названии UTF кодировки цифра обозначает минимальный размер, который может занимать один символ в битах. Т.е. UTF шрифты имеют динамическую размерность, а не фиксированную как в типе char.

Например, UTF-8 включает в себя символы следующих размеров:

- 1 байт или 8 бит для набора символов из ASCII
- 2 байта
- 3 байта
- 4 байта

В ходе реализации отображения эмодзи, описанные выше особенности и вызвали проблемы. Т.е. отображение символов в прошлой игре змейка производилось с помощью кодировки ASCII. Для змеи не важно в каком формате ее символ хранился в int или char. Главное, чтобы при выводе этого символа был указан правильный спецификатор %c.

Именно со спецификатором возникает две подпроблемы.

Первая, для UTF символов нет отдельного спецификатора как для char %c. Есть отдельный тип данных wchar_t. Но, как показало решение, можно обойтись и без "широкосимвольного" типа данных. Дело в том, что эмодзи, хранятся в памяти в виде последовательности байт, т.е. как строка или массив. При выводе нужно указывать именно этот спецификатор %s. Тут и возникает вторая подпроблема хранения эмодзи. Если ранее в структуре данных хвост или голову змеи мы помещали в char или int, то вторую подпроблему решаем с помощью указателей. Теперь в структуре лежит не символ, а указатель на эмодзи. Но, северный лис подкрался незаметно. И незаметность его заключается в том, что единицей измерения экрана ncurses является один символ. Переходим к проблеме многорядных эмодзи.

Многорядные эмодзи

Многорядные или мультибайтовые символы в UTF-8 для того, чтобы их отличить имеют специальное начало. Например, если поток символов имеет начало 0xF0, то этот символ будет много-байтовым. Было просто на бумаге, да забыли про овраги. Как было написано ранее, в СИ эмодзи хранятся в виде строки, а как известно она заканчивается нулевым символом \0. Т.е. если символ занимает один байт, то внутри строки он будет двумя байтами (символ + \0).

По началу, использовал UTF-8 с 4 байтовыми эмодзи, все работало хорошо. Красивый коптер с классными тележками бороздил просторы поля, до тех пор пока движение не начиналось вправо. Как только движение шло на право, весь

длинный состав из тележек и дрона, превращался в одну эмодзи. Долго, не мог понять причину. Перелопатил алгоритм движения в право. Но, это результата не дало. Как итог, нашел причину проблемы в многорядности эмодзи. По факту 4-байтовый эмодзи + \0 занимает 5 байт, т.е для его отображения нужно два ряда или две ячейки на экране. Т.к. письмо у нас слева на право, то по направлению письма он и перераспределяется и последний символ тележки перекрывает все предыдущие именно при движении направо. Решилось просто, использовал только эмодзи размерностью 3 байта. Да, их стало меньше и может быть не такие красивые и красочные. Но, в трех-байтовых, все ещ можно найти изображения коптеров и роботов, а также некое подобие вагончиков. Жаль, 4-байтовя тележка была очень похожа на Ашановскую)

Ссылка на трех-байтовые эмодзи:

https://design215.com/toolbox/utf8-3byte-characters.php

Исчезновение еды

При поедании первого элемента еды весь список пропадал, что было вызвано не правильной реализацией. Т.к. головной элемент списка является и указателем на всю последовательность его элементов, то при удалении первого стирается и последний. Проблема решена переопределением указателя на второй элемент списка.

Описание программы

В соответствии с ТЗ в АСУ "Сельхоздрон" реализован следующий функционал:

1. Дрон может перемещаться в плоскости. Перемещением дрона можно управлять вручную.
2. Дрон может определять границы тыквенного поля, эти границы ограничивают его перемещение.
3. Дрон может обнаруживать зрелые тыквы и собирать их в тележки для сбора.
4. Программа может отслеживать поведение целевых объектов:

- появление объекта на карте – обнаружение зрелой тыквы;
- удаление объекта с карты – зрелая тыква собрана дроном и больше не отображается на карте;

- обновление карты – сборка урожая может происходить с некоторой периодичностью, при обновлении карты на ней появляются новые зрелые тыквы.

5. Программа отслеживает количество собранного урожая. Урожай собирается в тележки, которые за собой возит дрон. Количество прикреплённых тележек для тыкв не ограничено. При сборке тыквы длина цепочки тележек увеличивается на 1.
6. Программа дрона уведомляет пользователя об аварийной ситуации: начало цепочки тележек с собранными тыквами столкнулось с концом.
7. Дрон имеет режим автопилота: искусственный интеллект управляет дроном по заданному маршруту.
8. Сборку урожая можно проводить несколькими дронами одновременно.

Исходный код программы

Отдельная ветка проекта drone, только код АСУ

Релизный файл содержащий только код АСУ

Много кода в составе всех заданий по курсу

Описание файлов программы

Проект АСУ состоит из 14 файлов:

- ai.c
- checks.c
- colors.c
- drone.c
- drone.h
- help.c
- inits.c
- Makefile
- maps.c
- menu.c
- moves.c

- prints.c
- queue.c
- README.md

Модуль ai.c

Содержит функцию ai(), реализующую простейший ИИ для работы дрона в режиме автоматического сбора тыкв.

Модуль checks.c

Функции проверки состояний:

- checkDirection() - проверка возможности перемещения дрона в заданное ему направление.
- checkFood() - проверка наличия еды в ячейке "головы" дрона
- isCrush() - проверка того что дрон не столкнулся с собой, по факту проверяет зависание дрона, т.е. движется ли они или нет.

Модуль colors.c

setColor(int color) - задает 7 парных палитр цветов.

Модуль drone.c

Основной модуль АСУ, содержит точку входа main(), которая определяет основной алгоритм работы АСУ.

Модуль drone.h

Заголовочный файл, содержит типы данных и прототипы функций, которые используются в модуле drone.c

Модуль help.c

- startHelp() - функция отображения краткой справки.

Модуль inits.c

Функции инициализации объектов на основе структурных типов данных:

- initDrone() - функция инициализации дрона
- initFood() - массовая генерация еды заданного типа и координатами через параметры.
- initStops() - функция добавления координат препятствий в стоп-лист

Модуль Makefile

Сборочный файла для утилиты make

Модуль maps.c

Функции создания карты объектов:

- mapPlots() - функция создания карты побегов
- mapPumpkins() - функция создания карты тыкв
- mapSpaces() - функция создания карты пустых (буферных) ячеек, для сглаживания проблемы "широких" эмодзи
- mapSprouts() - функция создания карты цветков
- mapWall() - функция создания карты вертикальных линий полей
- mapBottom() - функция создания карты нижних линий полей
- mapRoof() - функция создания карты верхних линий полей

Модуль menu.c

- startMenu() - функция отображения главного меню

Модуль moves.c

Функции перемещения дрона:

- right() - функция перемещения дрона вправо
- left() - функция перемещения дрона влево
- up() - функция перемещения дрона вверх
- down() - функция перемещения дрона вниз
- crawl() - функция перемещения в зависимости от нажатой клавиши, если нажатия не было, то движение происходит по последнему заданному направлению.

Модуль prints.c

printObject() - функция вывода переданного в параметр объекта на экран
printScore() - подсчет элементов в объекте и вывод на экран результата, по факту не используется. Нужна для вывода разноцветного результата для каждого дрона.

Модуль queue.c

Функции работы со списками:

- enqueue() - функция создания списков по принципу очереди
- concatQueues() - объединение двух списков в один, нужно для создания больших структур, например список всех тыкв на всех полях.
- iconQueues() - замена иконок во всем списке на заданную
- countQueues() - подсчет элементов в списке
- copyQueue() - копирование очереди, по факту не используется, предполагалось использовать вместо стоп-листа.

README.md

Данный файл, являющийся маркдаун копией курсового проекта, служит для отображения расширенной справкой в АСУ, вызываемой с помощью пункта 8 главного меню.

Описание и демонстрация работы программы

ЛЕГКИЙ ДЕМОНСТРАЦИОННЫЙ ВИДЕО РОЛИК 4.8 МегаБайта

При запуске АСУ в консоли терминала отображается главное меню:

```
АСУ Сельхоздрон!

Enter digit 0 - 9 to choose menu item:
0. Show or return to this menu
1. HELP and short description of the game
2. Start single player round
3. Start round with two players
4. Start round with computer vs player
5. Start round with computer vs computer
6. Set Anaconda color
7. Set Cobra color
8. Open help: README.md
9. Exit the game

Use ESC to exit help, or stop round
Use P to pause round, then press WASD to continue
```

При нажатии на клавиатуре соответствующей меню цифре, запускается нужный режим работы АСУ.

В видеоролике, демонстрируются все режимы работы АСУ.

Ролик начинается с главного меню, далее по порядку, происходит короткая демонстрация каждого режима.

Далее даются пояснения по ролику с привязкой по времени (тайм-аут):

- (00:02) Демонстрация короткой инструкции, можно нажать на видео-паузу и ознакомиться.
- (00:07) Демонстрация режима одного оператора, при нажатии ESC высвечивается информация по собранной тыкве и есть возможность вернуться в меню, либо выйти из программы.
- (00:22) Еще одна демонстрация режима одного оператора, обратите внимание на нижнюю часть экрана, при столкновении с объектами, такими

как линии поля, появляется значок коллизии и исчезает, т.к. критической остановки не произошло и движение дрона продолжается
- (00:42) Режим работы двух операторов, управлением дроном № 1 с помощью стрелок, дроном № 2 с помощью WASD. Что не удобно для управления одним человеком.
- (01:12) Режим работы оператора с компьютером. - (01:22) Дрон компьютера, вошел в жесткую коллизию, что привело к его остановке, о чем снизу сообщает сигнал со значком "кирпича".
- (01:33) Режим групповой работы четырех дронов под управлением ИИ. Обратите внимание на скорость принятия решений и реагирования. У человека, так быстро и ловко обходить препятствия не получиться. За каких-то 28 секунд весь урожай собран. На 26 секунде алгоритм обнаруживает ближайшую спелую тыкву на соседним поле, но принимает решение продолжить сбор на своем участке.
- (02:10) Выбор цвета для дрона №1
- (02:19) Выбор цвета для дрона №2
- (02:35) Режим работы оператора с человеком, видим что дрон человека теперь красного цвета.
- (02:57) Демонстрация работы отображение README.md файла с помощью АСУ, с возможностью листания вниз.

Сборка программы утилитой make

Т.к. в одном репозитории данного курса лежит много проектом, для уменьшения размера была создана отдельная ветка drone, которую можно скачать одним архивным файлом:

```
wget wget https://github.com/Kirill-Geek/Bit_Cu/archive/refs/heads/
Dz_7.zip
```

Полученный файло нужно распаковать с помощью команды unzip:

```
unzip drone.zip
```

Перейти в распакованную директорию:

```
cd c2-drone/07.Tasks/
```

И выполнить команду make

```
make
```

АСУ автоматически соберется при наличии необходимых зависимостей и библиотек упомянутых ранее. После успешной компиляции произойдет автоматический запуск АСУ.

Makefile

Makefile поддерживает три режима работы:

- сборка проекта
- пересборка
- очистка проекта

Сборка проекта

Для сборки проекта достаточно выполнить команду make в папке проекта:

```
make
```

Если исходный текст не изменялся, то проект повторно собираться не будет и сразу запуститься АСУ.

Пересборка проекта

В случае наличия изменений в исходном коде, при запуску утилиты make произойдет их проверка и соответствущие объектные файлы будут пресобраны

Очистка проекта

Для очистки проекта от бинарных файлов необходимо выполнить команду в папке проекта:

```
make clean
```

Все бинарные файлы будут удалены, и останутся только исходники АСУ.