MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION

"NOVOSIBIRSK NATIONAL RESEARCH UNIVERSITY

STATE UNIVERSITY"   
(NOVOSIBIRSK STATE UNIVERSITY, NSU)

15.03.06 - Mechatronics and Robotics

***OR***

Focus (profile): Artificial intelligence

09.03.01 - Informatics and Computer Engineering

Focus (profile): Software Engineering and Computer Science

**TERM PAPER**

**Authors**

Job topic:

**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"**

Novosibirsk, 2023

Table of contents

[Introduction](#_Toc103521432)

[1. Problem statement](#_Toc103521433)

[2. Analogues](#_Toc103521434)

[3. Hardware](#_Toc103521435)

[4. Software part](#_Toc103521436)

5. User Manual ……………..…………………………………………………………

[Conclusion](#_Toc103521437)

***Comments:***

Содержание пояснительной записки к курсовому проекту приведено (см. выше) «базовое». Эти разделы в пояснительной записке должны быть.

Но количество разделов (глав) может быть любым (можно проявить фантазию).

Главная цель пояснительной записки:

1. Показать, что ваш команда сделала;
2. Привести подробное описание «Hardware»;

- для проектов А, B, C – оригинальность (что-то дополнили, сделали по-другому, не так как описано в техническом задании, и т.п.);

1. Описание «Software»;
2. Описание взаимодействия Software и Hardware;
3. Заключение – это итог всей работе: кратко описываете, что ваша группа выполнила, выводы.

Пояснительная записка должна быть написана так, чтобы посторонний человек (специалист) мог «поиграть» в игру и разобраться в работе схем, программы.

Игра «пинбол» представляет из себя поле, на котором расставлены стены и другие различные препятствия. На это поле запускается мяч, на который действует гравитация, и он активно отскакивает стен. Задача игрока управлять дверцами внизу поля, и удерживать мяч на поле как можно дольше. Игроку начисляются очки за каждый удар мяча о стенку.

В нашей реализации игры есть только стены 4 видов: горизонтальные, вертикальные, и 2 вида диагональных под углом 45. Дверцы, которыми управляет игрок не двигаются, а включаются и выключаются. Они управляются 2 отдельными кнопками, поэтому одновременно может быть закрыта только одна дверца.

Hardware

Для того чтобы такая игра выглядела красиво нам понадобилось использовать расширенный экран 64\*64 пикселя, собранный из 4 стандартных экранов. Для оптимального вывода во все 128 столбцов к каждому из экранов приставлены запоминающие блоки, которые каждый такт обновляют только 1 из рядов, оставляя остальные в предыдущем запомненном состоянии. Таким образом полная отрисовка экрана происходит за 32 такта.

Сначала мы сделали блок, который преобразует 1 стену в пиксельное изображение, и накладывает получившиеся изображения друг на друга. Такой алгоритм рисует 1 стену за 32 такта, кроме того, из-за сложности вычислений сами такты становятся очень медленными. Поэтому мы решили, что раз все стены в игре не двигаются, то мы можем 1 раз запустить алгоритм, запомнить получившееся состояние экранов, и после этого загружать его из RAM.

В итоге графическая часть состоит из 3 основных частей.

Загрузчик –

Загружает все стены из ROM и преобразует их в пиксельное изображение 32\*32. Каждый загрузчик отвечает ровно за 1 экран, кроме того, все стены обязаны существовать в пределах одного экрана.

Обработчик –

Перебирает столбцы, и дорисовывает необходимые детали: мяч и дверцы, после чего передаёт их на экраны. Работает одновременно с 4 экранами.

Запоминающее устройство –

Хранит тридцать два 32-битных числа - столбцы для каждого экрана. По сути, представляет из себя 32 регистра. Принимает на вход обновленный столбец и его номер. Отличие от обычного RAM в том, что он имеет 32 одновременно активных выхода, чтобы поддерживать изображение на экранах.

Изначально все стены хранятся в 4 ROM, в каждом до 8 стен. И дверцы хранятся в 2 отдельных константах. Каждая стена – это 26 бит. Первые 2 – это направление стены. Далее 3 восьмибитных числа: координата х, координата у и длина. Причём координаты всегда считаются от нижнего левого угла стены. А длина повернутой на 45 стены кажется в sqrt(2) раз больше, потому что в программе хранится именно количество её пикселей.

// возможно надо описать алгоритм перевода стен в пиксели, но пока что пофиг

На самом деле все координаты и длины стен не больше 5 бит (так как они существуют внутри экранов), но для удобства мы храним их так.

Для всех вычислений с мячом используется отдельный блок BALL.

Мяч хранится как одно 32-битное число. По смыслу это 4 восьмибитных числа: координаты x, y и скорость по осям x и y. Первые 6 бит координат мяча считаются целой частью, а остальные 2 – дробная часть, для большей точности движения. Скорость имеет дробную часть в 5 бит.

//Может тут стоит уточнить, что целая часть – это координаты в пикселях

Сдвиг мяча и влияние гравитации также сделаны аппаратно. Сдвиг всего лишь прибавляет к текущим координатам скорость, сдвинутую на 3 бита вправо. А гравитация после перемещения мяча вычитает из его скорости по оси y единицу.

Между сдвигами нам необходимо проверить столкновения мяча со стенами. Для этого перебираются все стены, которые находятся с мячом на одном экране и данные о мяче и стене передаются в CDM-8. Когда программа заканчивает вычисления, мы узнаём произошло ли столкновение, и если произошло, то отражаем скорость мяча и увеличиваем счётчик очков на 1

// Здесь возможно надо написать о том, как изменяет скорость при столкновении с каждым из 4 типов стен.

Software //пока не переводи

Сначала программа ждёт флаг, который означает, что начался ввод.

// здесь надо написать что-то умное о системе, которую мы используем для ввода, насколько я понял из презентации их бывает 2 варианта и у нас более простой

После этого программа принимает 8 чисел: координаты и скорость мяча, координаты, длину и тип стены. При этом все дробные части уже отброшены битовыми сдвигами, а мяч выровнен в пределах экрана. Поэтому первые 3 бита во всех координатах равны 0, а у скорости 5 битов незначащие.

Далее для удобства рассмотрения всех 4 видов стен одним кодом мы «поворачиваем» все координаты, чтобы стена стала горизонтальной. Если тип0, то всё уже ок, если тип1, то просто поменять местами все x и y. Если тип2, то x2=x+y, y2=x-y и если тип3, то x2=x-y, y2=x+y.

// я не на 100% уверен, что не перепутал нигде знак, сейчас не хочу на 100% подробно это описывать

Для проверки пересечения используется не на 100% точный алгоритм (так как идеальный геометрический алгоритм пересечения 2 отрезков требует использовать деление).

1. Проверяем что x мяча находится между левым и правым концами стены (или равен одному из них) Если нет, то сразу отбрасываем этот случай. (это главная неточность, ведь он может переместиться по х так, что всё-таки пересечёт её)

// надо протестировать, мб надо делать доп. проверку после перемещения, чтобы избежать прохождение мяча сквозь стены. Но скорее всего и так сработает.

1. Считаем, выше или ниже мяч чем стена до перемещения и после перемещения.

Если он оказался по разные стороны, либо хоть раз совпал по y со стеной, то происходит столкновение.

Записываем в выходные данные 0 или 1 в зависимости от результата. Само отражение посчитает аппаратная часть.

После этого в следующий байт выхода ставим флаг, означающий конец работы программы.

Итог: программа работает неебически долго, потому что ей надо запуститься 8 раз, и каждый запуск довольно дорогой. Можно стен в каждом экране сделать по 5-6 штук, это ускорит заметно, но всё ещё недостаточно.