Региональная Конференция научно-технического творчества школьников «Лабиринты науки»

Секция «Информационно-телекоммуникационные технологии: программирование»

Разработка игровой программы «Новый Морской бой»

Автор: Смирнов Андрей Андреевич,
лицей №2 г. Рыбинска
10 «А» класс

Научный руководитель:
Аргов Дмитрий Игоревич
учитель информатики
лицея № 2

Рыбинск, 2023 г.

Содержание

Введение **3**
Основная часть **4**
Глава 1 «Разработка структур данных» 4
Глава 2 «Разработка расстановки кораблей» 6
Глава 3 «Разработка алгоритма для виртуального противника» 7
Глава 4 «Реализация визуальной части игры» 9
Выводы **10**
Источники информации **11**
Приложения **13**

Введение

В настоящее время игровая индустрия развивается очень быстро. С 2002 по 2020 годы объём рынка игр вырос в 8,7 раз, а в последние пять лет рост составлял порядка 11% в год. Количество проведённых в играх часов выросло в 12,5 раз. При этом Россия находится на 11 месте в Топе стран по объёмам рынка. Игровая индустрия нашей страны не превышает 2% от мирового объёма. Основными лидерами индустрии в мире уже давно стали США, Китай и Япония, а внутренний рынок России в сравнении этими странами, всё ещё не обладает настолько развитой культурой приобретения контента, тем более, цифрового.

Стоит отметить, однако, что наш рынок постепенно эволюционирует и развивается в этом направлении. Если переходить с соотношения на цифры — объём рынка в России составляет свыше 2 миллиардов долларов (или, проще говоря, свыше 160 миллиардов рублей) в 2021 году и при этом является одним из самых перспективных, как и масштабных регионов.

Как и в мире, основным лидером сейчас у нашего рынка являются мобильные игры, хотя традиционно российские игроки всё ещё отдают большее предпочтение ПК — 51% геймеров всё ещё используют именно эту платформу. Мобильный сегмент у нас тоже набирает популярность, подобравшись к цифре в 41,1%, ну и 8,8%, которые приходятся на консоли. К 2023 году аналитики прогнозируют, что мировой рынок игровой индустрии вырастет до 218 миллиардов долларов.

Но перед тем, как играть в игру необходимо её разработать. Для этого необходима команда игровых разработчиков (Gamedev-ов). Зарплата у разработчиков игр от 25000 до 150000 тыс. руб. Они должны придумать концепт игры и реализовать его. В команде есть люди отвечающие за визуальную и программную часть игры. В визуальную часть входят: визуальные эффекты, музыка и др. А в программную часть входит реализация «механики игры».

Чаще всего разные задачи выполняют разные люди, но иногда встречаются программисты энтузиасты, делающие это всё сами. Среди начинающих программистов таких достаточно много. Вот и я решил себя в этом попробовать. Так как у меня нет собственной команды все необходимые задачи я должен буду выполнить сам. Я попробую себя и в роли дизайнера игр и разработаю свою «механику игры».

Существует достаточно много жанров игр такие как: симуляторы, аркады, стратегии и другие. В тройку самых популярных игровых жанров вошли казуальные игры, головоломки и ролевые игры. Я решил выбрать игру в жанре стратегия. Среди стратегий я выбрал реплику на компьютер достаточно старой игры «Морской бой». Актуальность данного проекта заключается в том, что на данный момент практически каждый может себя попробовать в разработке игр.

Цели:

1. Получение опыты в разработке 2D-игр.
2. Разработка игровой программы «Морской бой»

Задачи:

1. Разработать структуру данных игрового поля и кораблей
2. Разработать алгоритм оптимальной расстановки кораблей
3. Разработать алгоритм работы виртуального противника
4. Реализоватьвизуальную часть игры

Основная часть

1.Разработка структур данных

Для начала необходимо определиться с правилами игры. Было введено новое правило: во время хода можно переставить один из своих кораблей, если эта перестановка не противоречит основным правилам по расстановке кораблей. Итак, игру можно разбить на такие этапы:

1. Подготовка игрового поля

2. Игровой процесс
3. Подведение итогов

Подготовка игрового поля.

В самом начале двое игроков должны расставить свои корабли на своём игровом поле корабли. Количество кораблей определено: 1 «четырёхпалубный», 2 «трёхпалубных», 3 «двухпалубных» и 4 «однопалубных». Корабли представляют собой непрерывную линию длинной в количество палуб (пример: «четырёхпалубный» - 4 клетки). Игровое поле представляет собой поле размером 10х10 клеток. При расстановке корабли не должны соприкасаться (Приложение 1, Приложение 2).

Игровой процесс

Далее идёт основная «игровая» часть. Во время хода один из игроков может передвинуть один корабль на одну позицию вверх, вниз, вправо или вниз, далее он выбирает клетку противника и производит выстрел. Если выстрел оказался удачным, то есть игрок попал в корабль противника ход остается у игрока, а если выстрел оказался неудачным, то ход переходит противнику. Игра продолжается пока все корабли одного из оппонентов не будут выведены из строя.

Подведение итогов

В конце, когда все корабли одного из оппонентов необходимо подвести итоги. За каждую подбитую палубу начисляется балл. Максимальное количество баллов – 20.

Теперь необходимо это представить в программе. Игровое поле представляет собой матрицу 12х12клеток целочисленного типа, такие размеры были выбраны чтобы не было проблем с границами поля. Клетки пронумерованы от 1 до 10 сверху вниз, и от A до I слева направо. Для кораблейбылсозданспециальный массив и новый тип из таких характеристик как: его начальные координаты, его направление, его длина (или количество палуб), а также массив состояний каждой палубы. Он представляет собой

массив от 0 до длины корабля минус один логического типа. Если палуба подбита тогда в ячейке с номером палубы минус один значение становится «правда», изначально ячейке хранится значение «ложь».

2.Разработка расстановки кораблей

Для данного этапа было необходимо было создать курсор. Для него был создан отдельный тип, куда входили такие параметры как, координаты курсора, длина (тип) курсора, направление курсора. Перемещение курсора осуществляется с помощью клавиш вверх, вниз, вправо, влево. Изменение направления курсора осуществляется с помощью клавиши «L» поворот на 90° против часовой стрелки, а с помощью клавиши «R» поворот на 90° по часовой стрелке.

Во время перемещения необходимо проверять выходят ли за границы координаты курсора не только начальные, но и конечные. Для этого была создана процедура, находящая конечные координаты курсора по начальным координатам и направлению курсора. Также во время поворота необходимо проверять выходят ли за границы координаты (Приложение 11).

Расстановка корабля осуществляется по клавише «Enter». После нажатия вызывается процедура PutShip. В параметры данной процедуры входят такие параметры как: поле игрока, поле виртуального противника, количество кораблей игрока, количество кораблей виртуального противника, режим расстановки, текущее состояние курсора, то есть его координаты, направление, и длина.

В начале процедуры необходимо проверить, что текущее количество кораблей не больше 10 и что можно поставить в текущие координаты корабль. Далее вызывается процедура NewShip. В данной процедуре в ячейку массива кораблей добавляется новый корабль. После того как создан новый корабль в ячейки, соответствующие расположению корабля, присваивается значение равное текущему количеству кораблей. В дальнейшем, если происходит попадание по клетке тогда мы с легкостью сможем обратиться к кораблю со значением в клетке минус один. После того как в матрицу уже

добавлен корабль необходимо сделать границу вокруг корабля, чтобы нельзя было поставить близко другие корабли, то есть сохранялось правило расстановки кораблей. После того как корабль был внесён и в матрицу, и в массив кораблей необходимо изменить длину курсора в соответствии 1 корабль – длина 4 клетки (четырёхпалубный корабль), от 2 до 3 – длина 3 клетки (трёхпалубный корабль), от 3 до 5–длина 2 клетки (двухпалубный корабль), от 6 до 10 – длина 1 клетка (однопалубный корабль).

На данный момент рассмотрена расстановка кораблей для игрока, теперь необходимо рассмотреть расстановку кораблей для виртуального противника. В первую очередь рассмотрим выгодные тактики расстановки кораблей. Рассмотрим эту тактику. Когда корабль уничтожается вокруг него образуется, область в одну клетку, где не может быть корабля по правилам. Вся суть данной тактики заключается в том, что нужно минимизировать данные клетки, где не может быть кораблей. Для этого корабли необходимо расставить по углам и вдоль границ, также корабли можно располагать близко друг к другу (Приложение 4). На первый взгляд это замечательная тактика расстановки кораблей, но после нескольких партий игрок может понять данную тактику. Данную проблема была решена выбором в пользу случайной расстановки (Приложение 3). Расстановка кораблей виртуального противника аналогична расстановке кораблей у игрока за исключением выбора координат. Выбор координат и направления осуществляется за счёт функции, которая выдаёт случайное значение в определённом диапазоне. Во время генерации проверяется, что основные и конечные координаты не выходят за границы.

3.Разработка алгоритма для виртуального противника

Теперь необходимо переходить к реализации игрового процесса. После того как игрок расставил корабли курсор теперь расположен на поле противника. При нажатии клавиши Enter производится выстрел. Если на месте, где был расположен курсор значение матрицы от 1 до 10 значит выстрел был «удачным» и ход остается у игрока (Приложение 7). Далее вычисляется

расстояние от клетки, по которой был произведён выстрел до основных координат корабля с индексом равному значению этой клетке. После этого в массив жизней корабля с индексом расстояния присваивается значение «правда», означающее, что данная палуба подбита. Даже если выстрел оказался «неудачным» этой клетке присваиваться значение 254 означающее, что в данную клетку стрелять уже нельзя (Приложение 5, *Приложение 6*). Также если игрок не попал в корабль ход переходит оппоненту.

А сейчас поговорим о самом интересном – алгоритме для виртуального противника (Приложение 9, Приложение 10). Самым сложным оказалось реализация перемещения кораблей. Для противника было создано две очереди: выстрелов и кораблей, которые необходимо переставить. В самом начале необходимо проверить очередь кораблей для перемещения. Если она не пустая, тогда пробуем переставить корабли. Также скажем, что для перемещения будут выбраны корабли, которые оказались ранены. Если очередь пустая, тогда проверяем все корабли на необходимость перестановки. Если даже после проверки корабли переставлять не надо, тогда переходим к выстрелу. В начале выстрела необходимо проверить очередь выстрелов. Если очередь не пустая, тогда достаём из неё координаты для очередного выстрела, пока клетка для выстрела не будет свободной. Если очередь пустая, тогда выбираем новую клетку по следующему алгоритму. В нём для каждой клетки подсчитывается вероятность нахождения в ней корабля, также учитывается текущее количество кораблей. После того как был произведён выстрел необходимо проверить, оказался ли он удачным. Удачным выстрелом будем считать выстрел, который повредил палубу или уничтожил корабль. Если выстрел удачный, проверятся уничтожен ли корабль этим выстрелом. В случае если это так очищается очередь, иначе соседние клетки помещаются в очередь и, если клетка ранее была выбрана из очереди задается приоритет направления по горизонтали или вертикали. Из-за возможности перемещений пришлось ввести исключение. Оно заключается в том, чтов случае попадания виртуальным противником в

клетку, где располагается подбитая палуба ход остаётся у противника. Временно снимается приоритет направлений и в очередь помещаются соседние клетки. Таким образом, виртуальный противник «не потеряет» корабль, если его переместили.

Немного о перемещении. Чтобы проверить можно ли переставить корабль, выполняется «тестовая» перестановка, в результате которой принимается решение о том, что можно или нельзя переставитькорабль. Теперь необходимо реализовать алгоритм. Для этого создаем обычную очередь и инициализируем её. Создаём процедуры необходимые для работы с ней (Приложение 12). Было решено ввести счет в игру. Для счета созданы специальные баллы, которые начисляются при удачном попадании. Привилегия первого выстрела принадлежит игроку. После того как все корабли одного оппонентов уничтожены выводится экран окончания, в котором можно увидеть выиграл ты или нет и какое количество очков ты набрал.

4.Релизация визуальной части игры

Основная механика игры разработана, а это значит пора переходить к визуальной части. Для начала определимся что нам для этого необходимо. Во-первых, это изображения кораблей, во-вторых, это изображение для заднего фона. В силу того, что проект написан на PascalABC.Net можно использовать изображения в формате jpg, png и bmp. Так как png более современный и поддерживает прозрачные пиксели для изображений был выбран именно этот формат. Все изображения были найдены в интернете и обработаны в программе Gimp.

Обработка представляла собой изменение размеров и поворот на 90° в ту или иную сторону. У кораблей может быть четыре направления, следовательно, должно быть четыре изображения: 0 – влево, 1 – вверх, 2 – вправо, 3 – вниз. Названия корабля необходимо зашифровать «sh_“тип корабля” _ “направление корабля”.png», например изображение с

однопалубным кораблём направленным влево будет выглядеть так «sh_1_0.png».

Для изображений кораблей был создан отдельный массив для более удобного доступа. Данный массив индексирован от 0 до 3 (направление) и от 1 до 4 (тип корабля). Изначально на задний фон выводилась статичная картинка волн, но было принято решение её «оживить». Для этого необходимо увеличить ширину изображения вдвое. Далее происходит перемещение изображения на некоторое количество пикселей. Таким образом происходит иллюзия перемещения волн. Курсор представляет полупрозрачные квадраты зелёного цвета, если в данную клетку(и) можно поставить корабль, или произвести выстрел. В клетках, по которым уже производились выстрелы и они оказались «неудачными» или для отображения клеток в которых корабли не могут быть расположены выводится изображение мины.

Также необходимо отображать, что палуба у корабля подбита. Для этого поверх корабля вводится изображения огня. Для огня было взято gifизображение огня и разбито на кадры. Для огня был создан специальный массив, где хранились покадрово изображения корабля. Для того чтобы игрок не видел, как располагаются корабли виртуального противника они выводятся после того, как полностью уничтожены (Приложение 8).

Теперь необходимо сделать экран приветствия и экран окончания. Для экрана приветствия было выбрано gifизображение плывущего корабля и разложено на кадры. Для экрана было выбрано статичное изображение. На экране выводится окончание боя такая, как: победил игрок или проиграл и какое количество баллов он набрал. В ходе тестирования игры была выявлена проблема того, что новый игрок не знает какие имеются органы управления и после экрана приветствия выводятся изображение «Инструкция по управлению». Также были добавлены звуковые эффекты такие как: звук печатающей машинки и фоновая музыка.

Выводы

В ходе выполнения этой работы я получил интересный опыт и практические навыки в разработке 2D-игр, а также я

1. Разработал структуру данных для игрового поля и кораблей
2. Расстановку кораблей
3. Алгоритм для виртуального противника
4. Реализовал визуальную часть игры

И в итоге я смог создать свою игру «Морской бой».

Также я провел тестирование моей игры и выявил плюсы и минусы моей игры.

Достоинства моей игры:

1. Современный дизайн.
2. Удобный интерфейс.
3. Для работы требуется всего лишь процессор.

Недостатки моей игры:

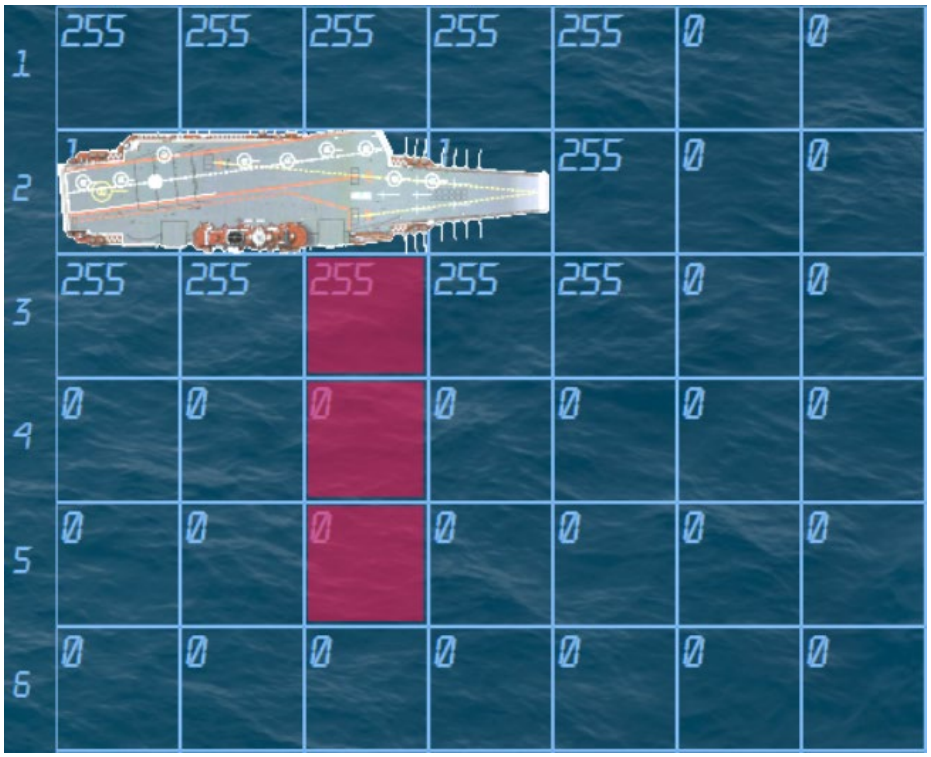
1. Нельзя выбирать сложность противника.
2. Одно разрешение экрана.

В будущем я собираюсь исправить недостатки и сделать свою игру лучше.

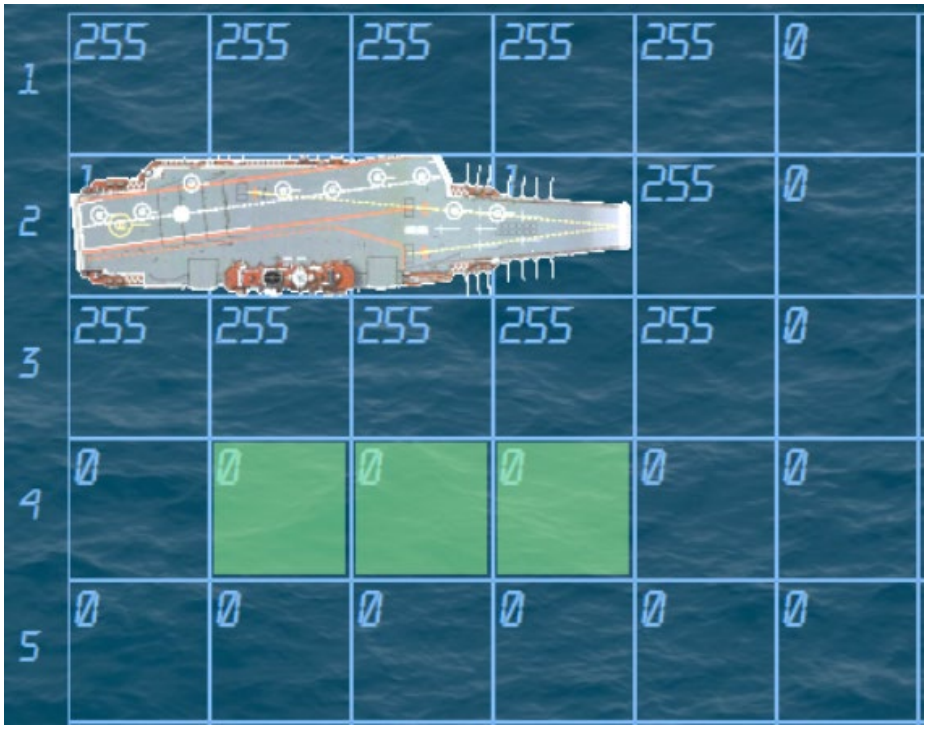
Источники информации

1. https://dtf.ru/gameindustry/888034-industriya-videoigr-v-rossii-i-mire
2. https://habr.com/ru/post/180995/
3. http://pascalabc.net/downloads/pabcnethelp/index.htm?page=PABCUnits/GraphABC/index.html
4. https://vacenko.ru/sistema-vyigrysha-v-igru-morskoj-boj/
5. https://ru.wikipedia.org/wiki/Морской_бой_(игра)
6. Аргов Д.И. презентация Структуры данных, Рыбинск 2016 г.

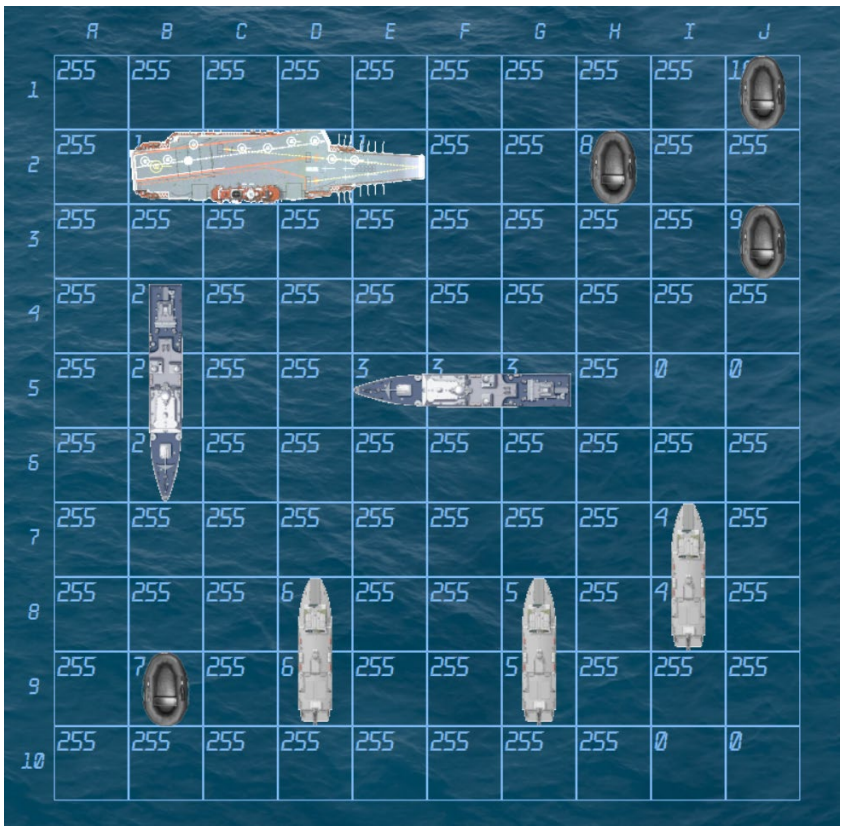
Приложения



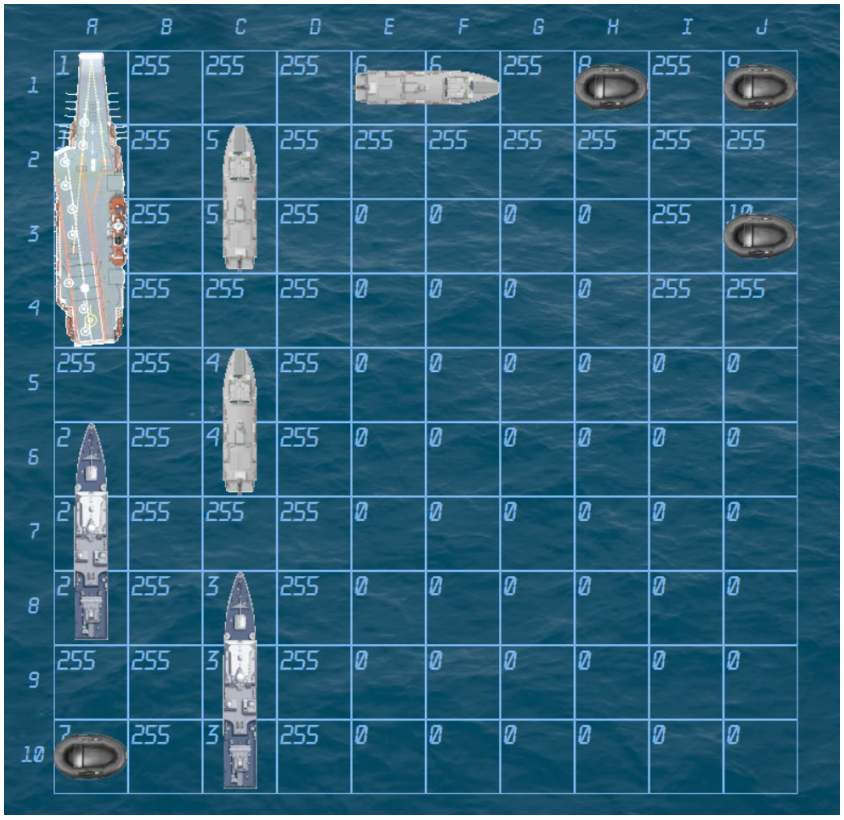
Приложение 1. «Пример, когда нельзя поставить корабль».



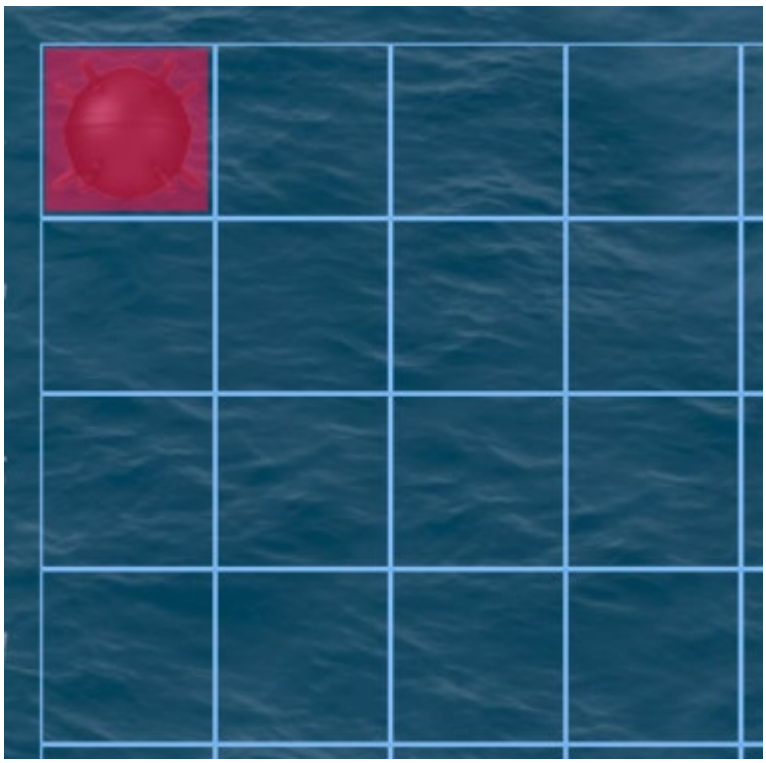
Приложение 2. «Пример, когда можно поставить корабль».



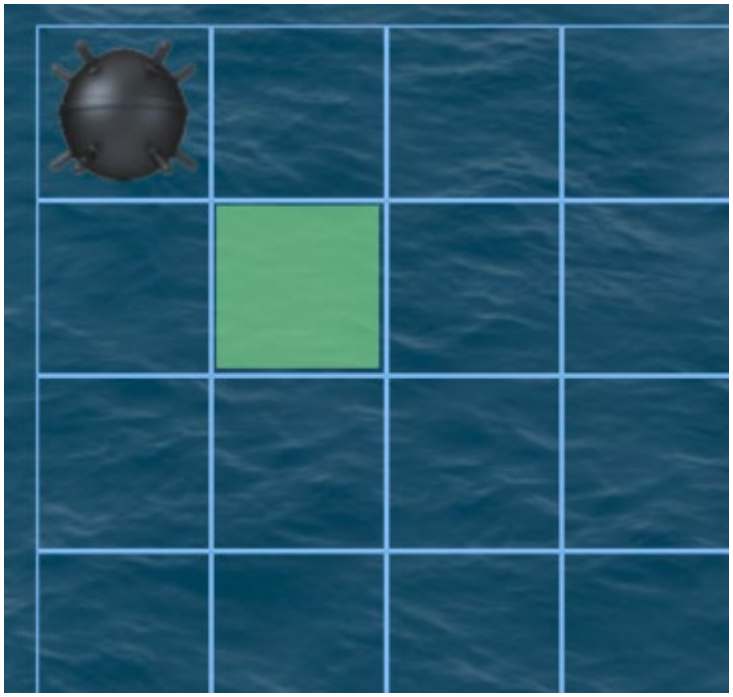
Приложение 3. «Пример расстановки кораблей».



Приложение 4. «Пример умной расстановки».



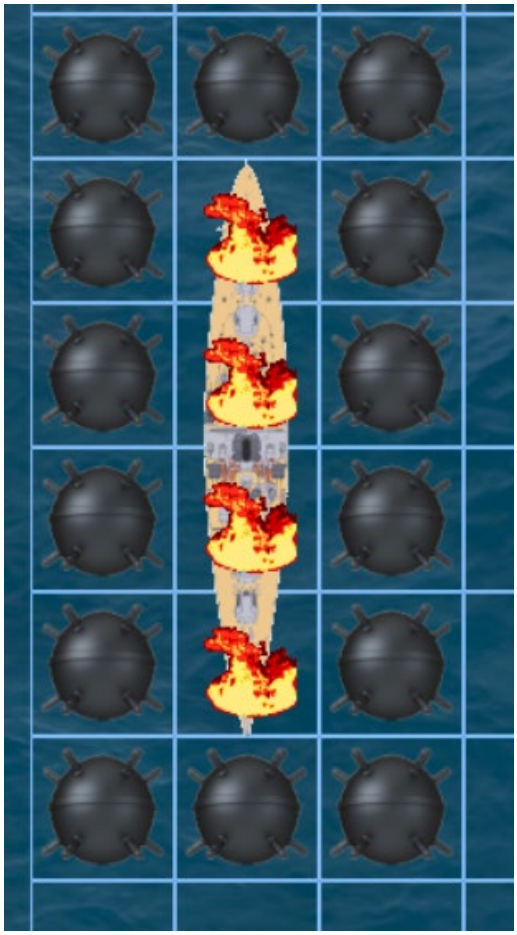
Приложение 5. «Пример показывающий, что нельзя стрелять в клетку куда был совершен выстрел»



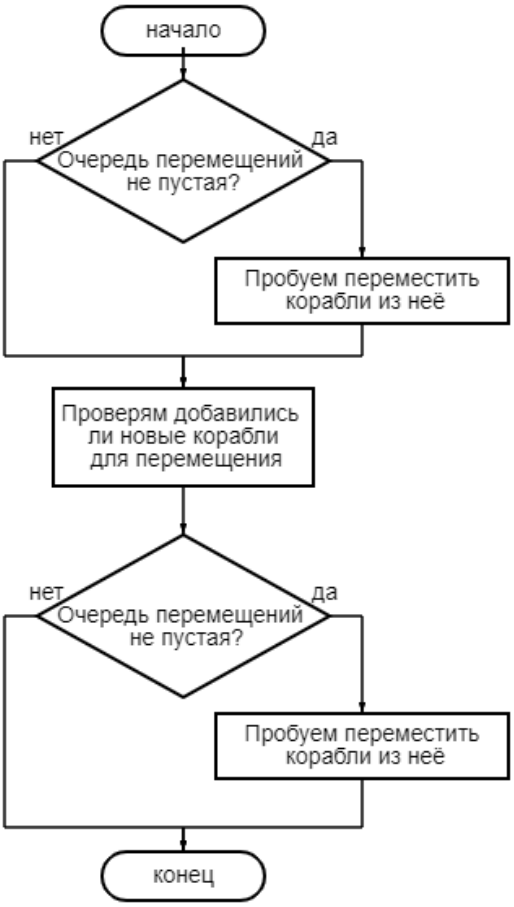
Приложение 6. «Пример показывающий, что можно стрелять в данную клетку».



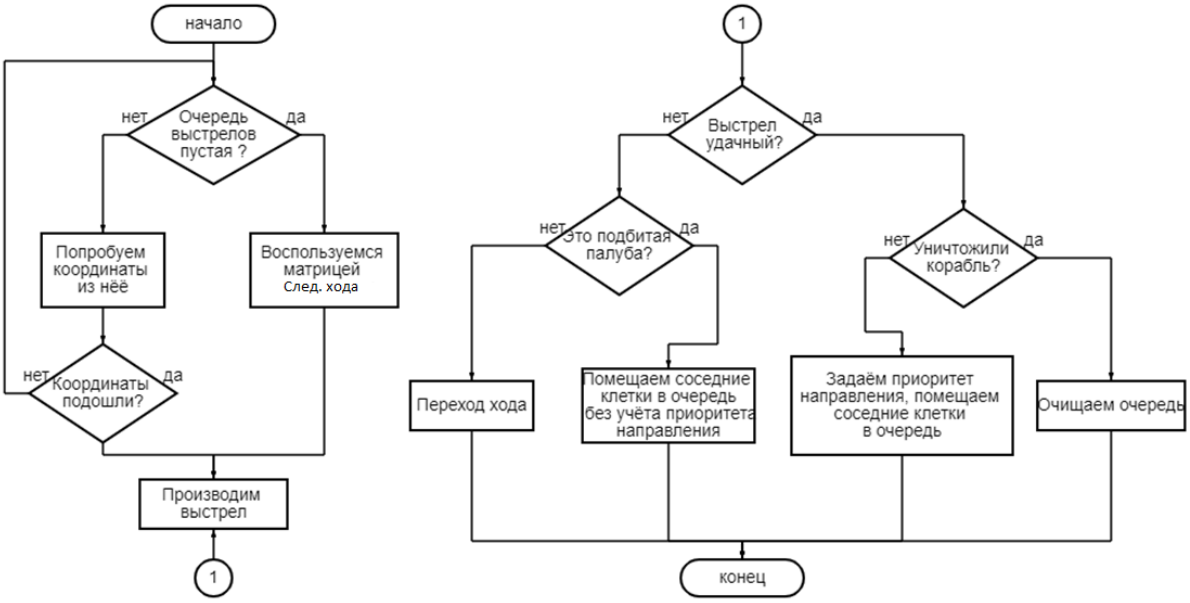
Приложение 7. «Подбитая клетка».



Приложение 8. «Убитый корабль».



Приложение 9. «Алгоритм выстрела виртуального противника, 1 часть».



Приложение 10. «Алгоритм выстрела виртуального противника, 2 часть».

```
procedure Rotare(direction: byte);
var
  xr, yr, r: integer;
begin
  r := cur.r;
  if direction = 1
  then
  begin
    if r = 3
      then r := 0
    else r := r + 1; {находим координаты после поворота}
    case r of
      3: begin xr := cur.x; yr := cur.y + cur.t end;
      2: begin xr := cur.x + cur.t; yr := cur.y end;
      1: begin xr := cur.x; yr := cur.y - cur.t end;
      0: begin xr := cur.x - cur.t; yr := cur.t end;
    end;
    if (xr >= 0) and (xr <= 11) and (yr >= 0) and (yr <= 11)
    then begin  {если координаты не за границами, тогда поворачиваем}
      if cur.r = 3
        then cur.r := 0
      else cur.r := cur.r + 1;
    end;
  end
  else begin
    begin
      if r = 0
        then r := 3
      else r := r - 1;
      case r of
        3: begin xr := cur.x; yr := cur.y + cur.t end;
        2: begin xr := cur.x + cur.t; yr := cur.y end;
        1: begin xr := cur.x; yr := cur.y - cur.t end;
        0: begin xr := cur.x - cur.t; yr := cur.t end;
      end;
      if (xr >= 0) and (xr <= 11) and (yr >= 0) and (yr <= 11)
      then begin
        if cur.r = 0
          then cur.r := 3
        else cur.r := cur.r - 1;
      end;
    end;
  end;
end;
```

Приложение 11. «Процедура поворота курсора».

```
procedure PutQ(x: keyt); {процедура помещающая новый объект в очередь}
begin
  if l < max
  then begin
    inc(xv);
    if xv > max then xv := 1;
    q[xv] := x;
    inc(l);
  end;
end;

function GetQ: keyt; {функция возвращающая первый объект из очереди}
var
  x: keyt;
begin
  if l > 0
  then begin
    x := q[g];
    inc(g);
    if g > max then g := 1;
    dec(l);
    getq := x;
  end
  else begin
    x.x := 0;
    x.y := 0;
    getq := x;
  end;
end;
```

Приложение 12. «Процедуры для работы с очередью».