Урок 7. Первая игра

Теперь мы готовы написать полноценную игру. Сюжет игры будет таков. Игрок должен уврачеваться от падающего на него шара и расстреливать его клавишей «пробел». При попадании шар разделяется на два шара с большей скоростью , но меньшим размерам. При повторном попадании получается то же самое. Если попасть по самому маленькому шару, то он исчезает. В учебном примере lesson7.pas , приведена только часть игры – когда при попадании по первому шару образуются два новых шара, и шары не наносят вред игроку.

Данные о шаре лучше хранить вместе и передавать их в функции, которые будут работать с этим шаром. Но как это осуществить? Для этого есть существует возможность объявлять произвольные типы данных. Перед блоком констант пишется оператор **type** после чего, на следующей строчке мы создаём наш тип данных TBatte\_Circle. Этот тип данных содержит все элементы, которым нам необходимы. Если данные одинаковы, то мы можем объявить новым типом данных массив. Например

TBatte\_Circle = **array**

означает, что при передачи параметра TBatte\_Circle будет передаваться весь массив. Если нам нужно использовать элементы разного типа, например Double и Integer , для этого используют **записи** . *Записи* создаются с помощью оператора **record :**

TBatte\_Circle = **record**

Format : Integer; // Первый,второй или третий размер.

Vx : Double;

Vy : Integer; // -1 или 1

X : Double;

Y : Double;

**end**;

Данный код означает, что создан тип данных , представляющий собой *запись* с элементами Format,Vx,Vy,X,Y .

Вызов функции с этим типом ничем не будет отличаться от вызова функции со стандартным типом. Например

Function NextXY(Batte\_Circle:TBatte\_Circle):Integer;

означает , что мы передали функции одну *запись*, содержащую вышеуказанные элементы. Доступ к любому элементу записи осуществляется через оператор «**.**» Например, код

Batte\_Circle.X := Batte\_Circle.X + Batte\_Circle.VX\*VelocityScale;

означает, что мы к элементу записи X прибавили элемент той же записи Vx, умноженный на константу VelocityScale.

Существует также ещё один стандартный тип, удобный для использование с операторами if. Это тип **Boolean**. Переменная или константа этого типа может принимать только два значения – истина или ложь. Если переменная B в момент странения if B примет значение «истина», то будет выполняться то что написанно следом за then . Например значение функция

**function** Intersect(Circle1,Circle2:TBatte\_Circle) : Boolean;

Проверяет, пересекаются ли две оружности. Если пересекаются, то на выходе выдаётся значение «истина». Иначе выдаётся значение «ложь».

Для улучшения вычислений внутри функции тоже можно применять **локальные переменные** преимущество перед обычными переменными заключается в том, что используя их мы никак не влияем на общее состояние программы. Это позволяет переносить код функции из одно программы в другую.

До этого мы рассматривали функции, которые возвращают на выходе какое-то обозначение. Бывают также функции которые не возвращают никакого значения. В Паскале принято обозначать такие функции оператором **procedure**

**procedure** KeyPress(key:integer);

означает, что функция KeyPress не возвращает никакого значения. Так как используется другой оператор, в большинстве справочников по языку Паскаль такие функции называются **процедурами**. Однако в других языках программирования подобный термин не применяется, и функции всегда называются функциями.

До этого мы передавали в функции константы. То есть не могли изменить в функции то, что мы ей передали. Если эту требуется, то параметр функции объявляется как переменная например

**procedure** NextXY(**var** Batte\_Circle:TBatte\_Circle);

Означает, что внутри функции можно изменять переменную Batte\_Circle .

Попробуйте убрать оператор var и убедитесь, что после этого все шары перестали двигаться, потому что теперь функции не может менять переданный ей параметр Batte\_Circle.

Кроме ранее изученного цикла for в данной программе применяется ещё один вид циклов. Repeat… until все , что находиться между двумя операциями выполняется столько раз , пока не настанет условие, когда цикл надо прекратить. В нашем случае это нажатие кнопки «Esc».

Также нам нужно изучить ещё ряд стандартный функции графического модуля.

procedure TextOut(x, y: integer; t: string)

Выводит текст t в прямоугольник, у которого верхняя левая точка имеет координаты (x, y).

procedure FillCircle(x, y, r: integer)

Заполняет окружность с координатами центра (x, y) и радиусом r текущим цветом заливки.

SetBrushColor(c: Color) устанавливает в качестве текущего цвета заливки цвет с.

onkeydown:= KeyPress;

означает, что если произошло нажатие клавиши, запускается функция KeyPress.

SetLength(i:Integer) - стандартная функция pascalABC.net , определяет длинну массива.

sleep(i:Integer) -стандартная функция pascalABC.net , останавливает выполнение программы на указанное количество миллисекунд.

Запустите игру. Управление осуществляется стрелками вправо-влево и клавишей пробел. Изучите код и выполните следующие задания.

1. В случае столкновения игрока и вражеского шара игра перезапускается. Выводиться текст, сколько раз игрок уже был убит.
2. В случае подпития вражеского шара выводиться текст, сколько раз он был подбит, а не как в текущей версии программы, что подбито 0 врагов.
3. Если подбит шар размера 2, то он разделяется на 2 шара размера 3.
4. Если подбит шар размера 3 , то он исчезает.
5. Если подбиты все вражеские шары, выводиться текст с поздравлением.

Заключение.

На этом завершается первая часть нашего курса. Окончивший первую часть данного курса может сделать простейшую игру. Рекомендуется не останавливаться на достигнутом, а модифицировать игру из урока или вовсе сделать игру со своим сюжетом. Для этого вам помогут следующие ссылки

<https://ru.wikibooks.org/wiki/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B2_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_GraphABC> - Введение в модуль GraphABC.

<http://pascalabc.net/primeri-programm/programmi-i-algoritmi-dlya-nachinaiuschich> - Примеры и алгоритмы для начинающих.

<http://pascalabc.net/downloads/pabcnethelp/index.htm> -более подробный справочник по работе с языком PascalAbc.net