Атака клонов

http://advantage.ucoz.ru
Автор: Stasundr

Вступление

Когда-то давно я изучал QBasic на уроках информатики. Помню, мы только прошли простейшие функции рисования: разноцветные прямоугольники и круги в режиме SCREEN 9. Вот тогда и началось веселье. Я рисовал круг за кругом, один в другом и все разными цветами. Код разрастался. Еще мне особенно нравилось, что скорость у компьютеров была очень маленькая, и он физически не мог нарисовать все эти круги сразу, а потому рисовал постепенно. Выглядело это впечатляюще. Сейчас я уже не помню, сколько строк было в том листинге, но что не меньше 200 – это точно. Тогда я еще не знал, что существуют такие замечательные вещи как циклы. Которых существует ни много, ни мало, четыре вида.

For

Первый цикл, который мы рассмотрим – это цикл **for**. Он имеет следующий синтаксис:

```
for ( i = 1; i < n; i += k ) {
        // тут мы что-то делаем.
}
```

Буквы и условие могут меняться, но структура остается прежней. Такой цикл выполнится до тех пор, пока i не станет равно n и с каждым проходом цикла i увеличивается на величину k. Как правило, k = 1. Этот вид цикла очень распространен, как и цикл **while**.

While

```
while ( i < n ) {
        // тут мы опять что-то делаем.
}
```

Таким образом, цикл будет выполняться до тех пор, пока условие в скобках не станет истиной. Очевидным отличием этого цикла от предыдущего, является то, что i не меняется автоматически, и мы сами должны следить за ним.

Until

Цикл **until** очень похож на цикл **while**. Отличие заключается в том, что в этом цикле условие проверяется после инструкций. Причем проверяется не истинность условия, а его ложность.

```
do {
        // и тут мы что-то делаем.
} until ( i > n )
```

И снова, заботу о переменной мы вынуждены взять на себя. Иначе цикл может стать бесконечным и игра попросту зависнет.

Repeat

Последний вид циклов можно назвать самым экономичным, потому что для него можно не заводить переменную.

```
repeat n {
        // снова что-то делаем. n раз.
}
```

Использование

Наконец, мы рассмотрели все виды циклов и теперь можно немного попрактиковаться в их применении. Очевидно, что разные виды циклов были придуманы не просто так. На то есть вполне конкретная причина. Этой причиной является удобство: в одних случаях удобен один вид, в других другой. Например, нам нужно заселить уровень некоторым количеством монстров. И это количество зависит от уровня сложности.

quantity не должна меняться.

```
quantity = mode * 15

// For
for ( i = 1; i < quantity; i += 1) instance_create( random( room_width), random( room_height ), o_monstr )

// While
i = quantity
while ( i > 0 ) {
   i -= 1
   instance_create( random( room_width), random( room_height ), o_monstr ) }

// Until
i = quantity
do {
   i -= 1
   instance_create( random( room_width), random( room_height ), o_monstr ) }
   until ( i == 0 )

//  Repeat
repeat quantity  instance_create( random( room_width), random( room_height ), o_monstr )
```

Не трудно заметить, что в данном случае самым удобным вариантом является последний. Однако так происходит далеко не всегда.

Итог

Циклы **while** и **until** хороши тем, что из них легко выйти (если условие соблюдено – цикл завершается), однако в них нужно следить, чтобы это самое условие когда-нибудь выполнилось. Цикл **for** сам следит за тем, чтобы завершится, однако он может выполнять лишнюю работу (если не использовать дополнительные конструкции выхода из цикла). Цикл **repeat** экономит память, но выполняется фиксированное заранее число раз. Зная все это, очень важно использовать именно тот тип цикла, который необходим в конкретной ситуации.

Удачи в нашем не легком деле, и до новых встреч, друзья!