УДК 372.862

ПОВЫШЕНИЕ МОТИВАЦИИ К ИЗУЧЕНИЮ ПРОГРАММИРОВАНИЯ ЧЕРЕЗ РАЗРАБОТКУ ИГРОВЫХ ПРОГРАММ

Лысенко Д.В.,
научный руководитель канд. физ.-мат. наук Дмитриев В.Л.
Стерлитамакский филиал Башкирского государственного университета

Все более значимую роль в человеческом обществе приобретают информационные технологии, которые проникли во все сферы нашей деятельности. В свою очередь, развиваются языки программирования, которые сейчас используются для решения разнообразных задач практической направленности. Поэтому содержание, особенности умственной деятельности обучающихся в процессе изучения программирования должны быть направлены на развитие системного, конструктивного, алгоритмического мышления, на формирование тех качеств и особенностей, которые позволят впоследствии молодым специалистам строить свою профессиональную деятельность наиболее эффективным образом.

В настоящее время при изучении программирования, как школьникам, так и студентам предлагается для решения большое разнообразие задач для отработки конкретных тем курса, причем такие задачи, как правило – чисто математические. Со временем решение только таких задач приводит к снижению интереса к программированию: решения этих задач не наглядны, а визуальный результат на фоне современных графических приложений и игр может полностью разочаровать.

Очень хорошим способом мотивировки изучения программирования является идея использования заданий, вызывающих наибольший интерес при работе с компьютером. Как правило, в качестве таких заданий выступают всевозможные задачи на разработку компьютерных игр. В этом случае в процессе обучения активно развиваются аналитическое, абстрактное и конструктивное мышление, способность к достижению цели, а элемент творчества является важнейшим компонентом для обеспечения мотивации обучающихся. Следует также отметить, что кроме традиционных конструкций структурного программирования в процессе работы над игровым проектом всегда присутствует возможность с достаточно высокой степенью наглядности освоить и воплотить принципы объектно-ориентированного программирования.

В качестве примера разработки игры рассмотрим задачу о нахождении пути в лабиринте. В такой игре участвует игрок и его противник (монстр), причем игрок может прокладывать путь, изменяя структуру лабиринта. Под изменением структуры лабиринта подразумевается разрушение стен лабиринта (взрываются гранатами) и возведение новых стен игроком. Также будем подразумевать, что на один шаг в лабиринте игрок тратит некоторое количество энергии. При перемещении монстра по лабиринту необходимо решить задачу поиска пути между игроком и монстром.

В рамках рассмотрения задачи о поиске кратчайшего пути между двумя точками в лабиринте наиболее целесообразно использование волнового алгоритма. Волновой алгоритм – это алгоритм, который позволяет найти минимальный путь в графе. В основе этого метода лежит алгоритм поиска в ширину.

Волновой алгоритм можно назвать одним из самых уникальных алгоритмов трассировки. Он позволяет сформировать путь (трассу) между двумя ключевыми точками (элементами) в любом лабиринте, если, конечно, задача является разрешимой. Разобьем лабиринт на отдельные клетки (ячейки). Каждой клетке присвоим одно из двух состояний: «пустая» и «препятствие». Также выберем клетки «начала» и «конца» пути.

Процесс поиска пути состоит из двух частей:

1. Из начального положения волна распространяется в 4-х направлениях. Элемент, в который пришла волна, создает новый фронт волны. Каждый из элементов первого фронта волны будет являться источником вторичной волны. Элементы второго фронта волны будут генерировать волну третьего фронта и т.д. Процесс формирования волн продолжается, пока не буде достигнут конечный элемент.

2. На втором этапе волнового алгоритма строится сама трасса. При этом его построение осуществляется от конечной ячейки к начальной.

Процесс распространения волны и пример построенного пути иллюстрирует рис. 1, на котором цифрами обозначены номера волн.

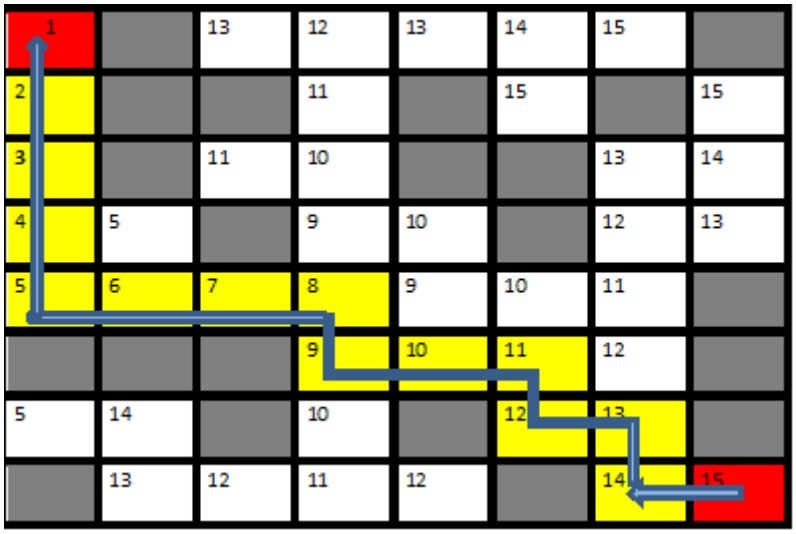

Рис. 1. Пример построенного пути в лабиринте.

Лабиринт будем хранить в двумерном массиве символьного типа. Для генерации лабиринта используем встроенный генератор случайных чисел – будем генерировать числа, например, от 0 до 99. При этом если сгенерированное число меньше 30, то соответствующий элемент массива будет хранить символ «#», означающий стену лабиринта (если брать другие числа, отличные от 30, можно регулировать проходимость лабиринта). Для задания игрока используем символ «И» (первоначально игрок располагается в лабиринте в позиции с заранее заданными координатами). Для завершения работы программы будем использовать кнопку *ESC*.

В разработанной программе реализована возможность игрока взрывать (разбивать) стены. Для взрыва стены использована клавиша «пробел». Здесь мы сталкиваемся с необходимостью запоминать, в каком направлении двигался игрок до того, как была нажата клавиша «пробел» – чтобы знать, с какой стороны производить взрыв стены (взрывать стену нужно в направлении движения игрока). Количество взрывов стен ограничивается – игрок обладает ограниченным количеством гранат (например, 10 штук). Если гранаты заканчиваются, игрок более не сможет разрушать стены. Добавлена также возможность сбора гранат, – в лабиринте размещены ящики с гранатами, например, по 5 единиц.

Рассмотренная игровая задача предполагает множество модификаций, и позволяет разработчику проявить свои творческие способности.

Как показывает практика, при работе над игровыми проектами мотивация обучающегося на протяжении всего времени разработки находится на очень высоком уровне. Поэтому такой подход способен внести реальный вклад в повышение эффективности обучения программированию, и методически верно вносить разработку компьютерных игр в процесс обучения.