ЛЕКСИЧЕСКИЙ АНАЛИЗАТОР ПРИ ГЕЙМИФИКАЦИИ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ

САВКИНА Анастасия Васильевна
кандидат технических наук, доцент
БАРАШИХИН Егор Александрович
студент
КЕПИЧ Денис Владимирович
студент
Мордовский государственный университет им. Н.П. Огарёва
г. Саранск, Россия

В статье кратко описывается реализация лексического анализатора для его использования в качестве обучающих последовательностей при геймификации в образовательном процессе, который создан на основе интегрированной среды разработки Visual Studio и кроссплатформенной среды создания компьютерных игр Unity, предназначенной для автоматизации многофункциональных кроссплатформенных приложений, работающих под Windows, Mac, Linux, iOS и Android.
Ключевые слова и словосочетания. Геймофикация; образование; Unity; NPC; лексический анализатор; функция, класс.

Изучение различных прикладных дисциплин, основой которых являются компьютерная графика, мультимедийные технологии, искусственный интеллект, машинное обучение занимают продолжительное время для освоения новых понятий и подходов. В этом случае геймификация, как альтернатива образовательной практики коренным образом поможет для продвижения интереса и коллективного вовлечения студентов в процесс обучения, обеспечив приобретение ими разумных уровней абстракции и логики, что в конечном итоге приведет к повышению уровня знаний и повышенного участия в освоении дисциплины [1; 3].

Для разработки последовательности выполнения лабораторных работ и создания системы автоматического оценивания необходимо сконструировать игровую модель с необходимым набором заданий на основе системы уровней, в которой для отработки навыков и повышения уровня знаний вводятся дополнительные условия с помощью списка типизированных заданий на каждом этапе. При этом необходимо продумать систему достижений, обеспечивающую систему оценок, благодаря которой у студентов будет познавательная мотивация к обучению, а также добавить кастомизацию основного героя обучения, то есть предусмотреть набор инструментов, которые будут востребованы обучающимися (игроками). В качестве основного программного продукта при разработке модели объекта обучения была использована платформа для разработки игр, объединяющая в себе инструментарий для создания точных трехмерных копий реальных объектов и пространств – Unity. Она содержит визуальный редактор, редактор кода, инструмент для написания скриптов – логики поведения объектов, который позволит быстро вносить изменения и в реальном времени оценивать результат. Платформа Unity предоставляет разработчику готовую физическую модель для взаимодействия между объектами виртуальной сцены, поэтому избавляет его от проработки поведения каждого из элементов. Встроенный в Unity физический движок включает законы, правила взаимодействия элементов сцены между собой и с окружающей средой [2]. Кроме этих возможностей в библиотеке много пресетов с разными настройками, которые можно загружать в проект и изменять. Богатая имитация физических явлений и объектов на основе частиц (атмосферные осадки, огонь, отражения) проводится с задействованием разработки от Nvidia PhysX и позволяет создавать запоминающиеся

образы. В состав физической стороны движка входит физика твёрдых и мягких тел. Система наследования свойств позволит дочерним предметам копировать свойства и поведение родительских, что значительно сокращает время на их тщательную разработку. Для создания обучающей игры предлагается карта с возможными заданиями и различными траекториями их выполнения (рисунок 1) и диалоговое окно с NPC (рисунок 2).

Рисунок 1. **Карта заданий**

Рисунок 2. **Диалоговое окно с NPC**

При выполнении студентом заданий очень важно реализовать лексический анализатор, который разбивает входную строку на лексемы - элементарные части языка программирования, такие как идентификаторы, ключевые слова, константы, операторы и разделители. Для этого в коде определен перечисляемый тип TokenType, который содержит все возможные типы лексем, определена структура Token, которая хранит тип и значение лексемы.

Основная логика работы лексического анализатора реализована в функции Tokenize, которая получает на вход строку, а на выходе возвращает список лексем, на которые была разбита входная строка.

```
public static List<Token> Tokenize(string code)
{
    List<Token> tokens = new List<Token>();
// Список лексем
```

Для разбиения строки на лексемы используется множество ключевых слов, операторов и разделителей, которые записаны в массивы keywords, operators и delimiters, соответственно.

```
string[] keywords = { "void", "int", "float",
"double", "char", "if", "else", "while", "for", "re-
turn" };
string[] operators = { "+", "-", "*", "/", "=",
"==", "!=", "<", "<=", ">", ">=" };
string[] delimiters = { ";", ",", "(", ")", "{",
"}" };
string currentToken = "";
```

В процессе работы функции Tokenize, строка обрабатывается посимвольно, при этом каждый символ добавляется в текущую лексему, которая формируется до тех пор, пока не будет встречен разделитель. Каждая лексема определяется ее типом, который зависит от первого символа лексемы. Если это цифра, то лексема является константой, иначе, если лексема является ключевым словом из массива keywords, то ее типом будет TOK_KEYWORD, иначе – TOK_IDENTIFIER. Если текущая лексема содержит оператор или разделитель, то она добавляется в список лексем, а затем очищается для обработки следующей лексемы. Далее,

реализуется сравнение двух списков лексем на схожесть – функция compareTokens, она проходит по всем лексемам первого списка и ищет совпадения во втором списке. После этого функция вычисляет процент совпадения и проверку на обязательные лексемы. После этого в строку code1 заносится весь текст из UI поля codeInputField1, который вводит студент. Применяем функцию Tokenize к коду студента формируем список tokens1. Затем выводим все токены в result1 и далее на UI codeInputField2. Сохраняем код студента во временное хранилище ResultStorage для дальнейшего вывода в файл. Проверяем код студента на соответствие заданным лексемам в функции CheckForTokens, и получаем на выходе булево значение true/false. Проверяем схожесть токенов в compareTokens и заносим найденный % соотношения в хранилище. Если код совпадает по лексемам, то выводим их в файл. В качестве временного хранилища данных был реализован класс ResulitStorage. Функция SaveResultData отвечает за создание файла с именем Result-Data.txt, в котором хранятся все данные, которые мы получили в ходе анализа кода студента для проверки – само задание, код студента, соответствие исходному коду в % и соответствие лексемам.

СПИСОК ЛИТЕРАТУРЫ

1. *Афонин В.В., Федосин С.А., Савкина А.В.* О повышении качества образовательных технологий на примере курса дисциплин направления «Информатика и вычислительная техника» // Образовательные технологии и общество (Educational Technology & Society). – Т. 22, № 1. – Казань, 2019. – С. 129-139.

2. *Корнилов А.В.* UNITY. Полное руководство, 2-е изд., ДМК Пресс, 2021. – 496 с.

3. *Савкина А.В., Черашева В.В.* Геймификация и сторителлинг в учебном процессе высшей школы // Столыпинский вестник. – № 11. – М., 2023. – С. 1-9.

LEXICAL ANALYZER FOR GAMIFICATION IN THE EDUCATIONAL PROCESS

SAVKINA Anastasiya Vasilievna
Candidate of Sciences in Technology, Associate Professor
BARASHIKHIN Egor Aleksandrovich
Student
KEPICH Denis Vladimirovich
Student
Mordovian State University named after N.P. Ogaryov
Saransk, Russia

The article briefly describes the implementation of a lexical analyzer for its use as training sequences for gamification in the educational process, which is based on the integrated development environment Visual Studio and the cross-platform environment for creating computer games Unity, designed to automate multi-functional cross-platform applications running on Windows, Mac, Linux, iOS and Android.

Keywords: Gamification; education; Unity; NPC; lexical analyzer; function, class.