Реализация образовательной среды программирования исполнителей на платформе REAL.NET

Егор Зайнуллин
Научный руководитель: доцент, к.т.н, Ю.В. Литвинов
Рецензент: педагог дополнительного образования М.М. Киселев

10 июня 2020 г.

Здравствуйте, уважаемая комиссия, здравствуйте, коллеги, меня зовут Егор Зайнуллин, тема мой выпускной работы «Реализация образовательной среды программирования исполнителей на платформе REAL.NET»

Мотивация

- Идея о создании единой системы обучения школьников
- Использование графических языков для обучения дошкольников и школьников младших классов
- Просьба от учителей 419 лицея разработать инструмент

С недавнего времени школьники сдают ЕГЭ, используя компьютер для программирования части заданий из экзамена, поэтому возникла идея о создании среды программирования, с помощью которой обучающиеся смогли бы изучать программирование, начиная с младших классов и визуального программирования, а заканчивая старшей школой уже с использованием более сложных текстовых языков. И использовать ее уже на самом экзамене. Это упрощает обучение, так как учащимся не надо знакомиться с новым программами для разработки. С этой идеей и просьбой к нам пришли преподаватели из 419 лицея. Для начала нужно разработать систему для обучения детей младшего школьного возраста.

Исполнители

Исполнитель

Абстрактный робот, выполняющий набор определенных команд

Исполнитель «Черепашка»

Умеет ездить вперед и назад, поворачивать на заданный угол, рисовать за собой линию.

Исполнитель «Робот»

Ездит по клетчатому прямоугольнику, у сторон клеток которого могут быть стенки. Ему запрещено проходить через стены.

Детям лучше преподавать в игровой форме. Для этой части обучения используются исполнители – абстрактные роботы, выполняющие определенные команды. Их использование позволяет наглядно показать выполнение алгоритма. В данной работе рассматриваются два исполнителя:
Исполнитель «Черепашка» умеет ездить вперед и назад, поворачивать и рисовать за собой линию.
Исполнитель «Робот» умеет ездить по клеткам прямоугольника со стенами и поворачивать на прямые углы. Проходить через стены ему нельзя.

Цель и задачи

Цель

Создание образовательной среды программирования исполнителей, то есть нескольких графических языков, интерпретаторов к ним и визуализаторов выполнения

- Рассмотреть существующие образовательные среды программирования
- Создать набор визуальных языков для программирования исполнителей
- Разработать архитектуру системы
- Реализовать исполнителя «Черепашка»
- Реализовать исполнителя «Робот»
- Провести тестирование и апробацию полученных средств

Цель проекта заключается в создании среды программирования исполнителей, то есть необходимо создать интерпретаторы и визуализаторы для них. В частности, для этого необходимо

- Выполнить обзор существующих сред
- Создать набор визуальных языков для программирования исполнителей
- Разработать архитектуру системы
- Реализовать исполнителя «Черепашка»
- Реализовать исполнителя «Робот» [Под реализацией понимается, что надо создать соответствующие интерпретаторы и визуализаторы]
- Провести апробацию полученных средств

Обзор существующих решений

Таблица: Сравнительная таблица

Название	Редактор	Интерфейс	Язык	Лицензия
FMSLogo	текстовой	английский	Лого, англ	бесплатная
ЛогоМиры	IDE	русский	Лого, русский	платная
КуМир	подсветка синтаксиса	русский	свой, русский	бесплатная

Рассмотрим существующие образовательные среды для программирования исполнителей.
Первая из них – это «FMSLogo». Популярен в англоязычных странах. Имеет простой текстовой редактор для написания программы. Полностью на английском. Работает с «Черепашкой» на языке «Лого». Является проектом с открытым исходным кодом.
Вторая среда – «ЛогоМиры». Является полноценной IDE. Также работает с «Черепашкой» и «Лого». Полностью на русском языке. Но является коммерческим проектом.
Третья рассмотренная среда – это «КуМир», проект с открытым исходным кодом, разработанный РАН. Редактор умеет подсвечивать синтаксис. Имеет свой собственный алгоритмический язык, поддерживает многих исполнителей, включая «Черепашку» и «Робота». Полностью на русском языке. Самими создателями рекомендован для преподавания, начиная со средней школы.

```
использовать Черепаха

алг
нач
. нц 2 раз
. . поднять хвост
. . вперед(100)
. . опустить хвост
. . влево(90)
. . вперед(100)
. кц
кон
```

Рис.: Алгоритмический язык, использующийся в КуМир

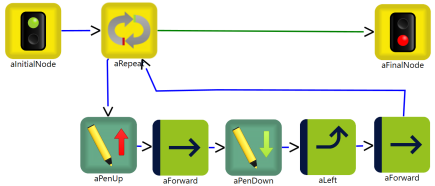


Рис.: Визуальный язык REAL.NET

На данном слайде изображено, как выглядят аналогичные программы для «Черепашки» на алгоритмическом языке «Кумир» и визуальном языке REAL.NET. По сравнению с языком «Лого» программа на языке «КуМир» длиннее, некоторые операторы состоят из двух слов. Визуальный язык представляет программу нагляднее.

Используемые технологии

- DSM-платформа REAL.NET
- Языки .NET: F#, C#
- Для графического интерфейса используется Windows Presentation Foundation(WPF)

В качестве основы используется DSM-платформа REAL.NET. Она предоставляет репозиторий для хранения моделей, а также редактор для написания программ. REAL.NET написан на языках .NET: F# и C#, так что сам проект также написан на них. Для графического интерфейса же используется WPF.

Требования к языку

- Поддержка циклов, условных выражений.
- Поддержка арифметических выражений, переменных.

Арифметические выражения

- Типы: целочисленные, вещественные, булевы и строковые, а также массивы над ними
- Бинарные операции: +, -, /, *, логические операторы, операторы сравнения
- Унарные операторы: - и отрицание
- Вызов функций
- Присваивание
- Инициализация массива

Каждый из создаваемых языков должен удовлетворять следующим требованиям

- Поддерживать циклы и условные операторы
- А также поддерживать арифметические выражения и переменные

Сами же арифметические выражения могут состоять из следующих конструкций

- Переменных и массивов
- Бинарных операторов, таких как: арифметических, логических, сравнения
- Унарных: - (взятие противоположного) и отрицания
- Вызовов функций
- Инициализаций массивов

В выражениях могут быть использованы такие типы, как: целочисленный и вещественный, булевый и строковый, а также массивы над указанными типами

Язык. Метамодель

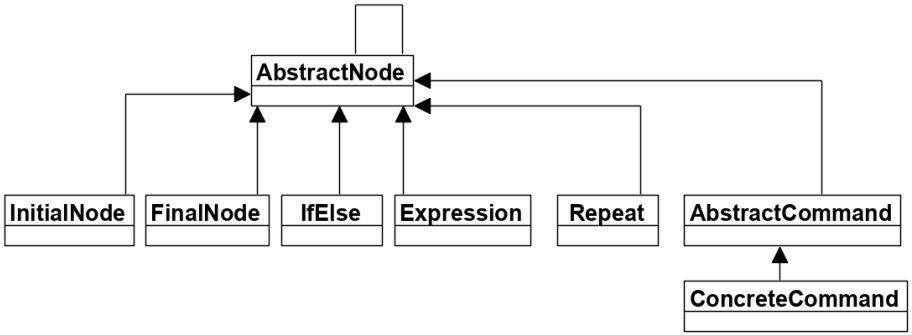
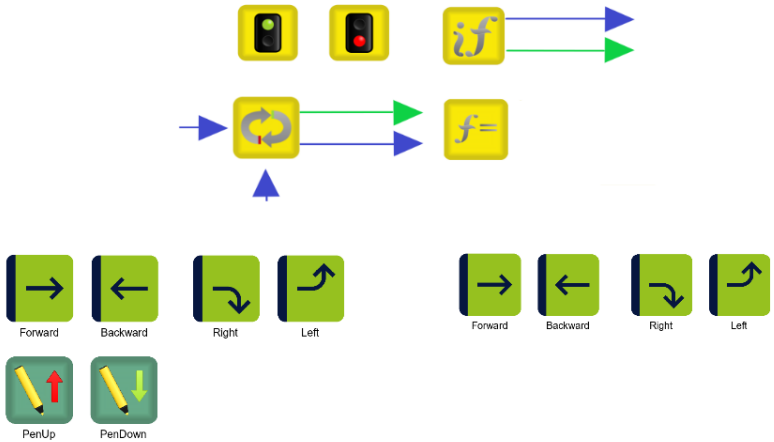


Рис.: Диаграмма классов языка

Визуальный язык описывается с помощью метамодели. Метамодель для языков исполнителей хранится в виде диаграммы классов. Она представлена на слайде. Все операторы наследуются от абстрактной вершины. Данная вершина имеет ассоциацию сама с собой, это значит, что ребро может быть проведено от любой вершины до любой другой. Непосредственно от нее наследуются общие конструкции для всех языков, а также абстрактная команда. Потомками же абстрактной команды являются специфичные для исполнителя команды.

Операторы языка

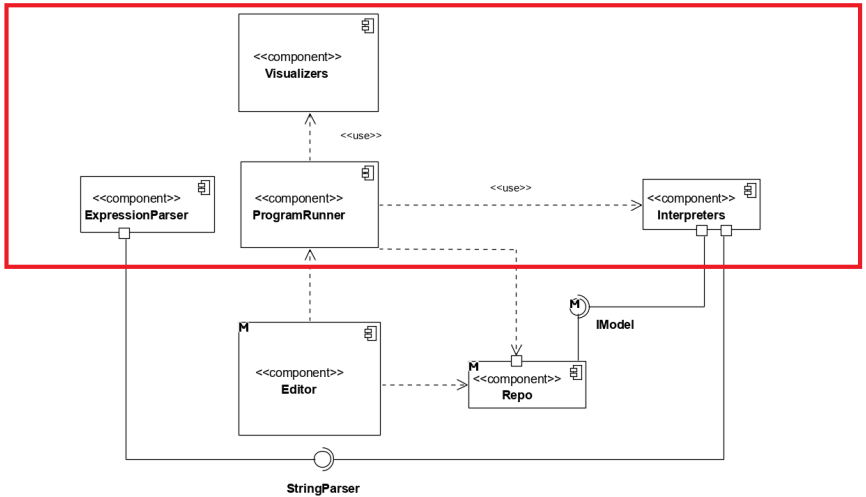


На данном слайде изображены общие конструкции

- Цикл, который повторяет свое тело указанное число раз
- Задание выражения, в котором определяются переменные
- Условное выражение, из которого, после определения истинности выражения, выходят по одному ребру, если данное выражение истинно, по другому – если ложно
- Также в виде светофора изображены начало и конец программы

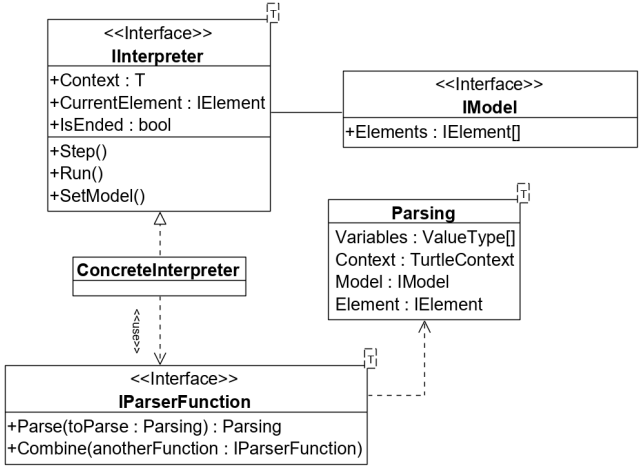
Здесь также изображены команды «Черепашки»: ехать вперед и назад на произвольное расстояние, повороты влево и вправо, опускание и поднятие пера – это нужно для рисования. На этом слайде есть и команды «Робота»: ехать на одну клетку вперед или назад, поворачивать под прямым углом.

Компоненты системы



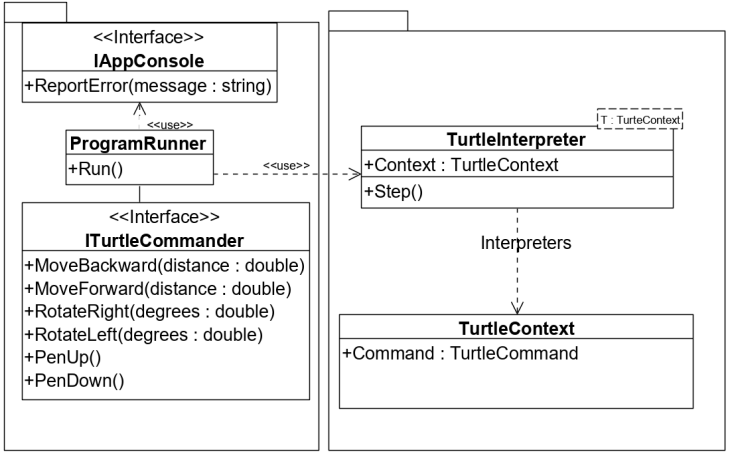
На данном слайде изображена диаграмма компонентов, которая позволяет понять, как работает вся система.
В редакторе рисуется модель. Когда пользователь запускает выполнение программы, ProgramRunner передает исполняемую модель интерпретатору, он последовательно проходит от начала программы до конца, выдавая на каждом шагу команды, которые следует выполнить исполнителю. Визуализатор их в свою очередь отображает.
Красным отображена часть, которой занимаюсь непосредственно я.

Интерпретатор



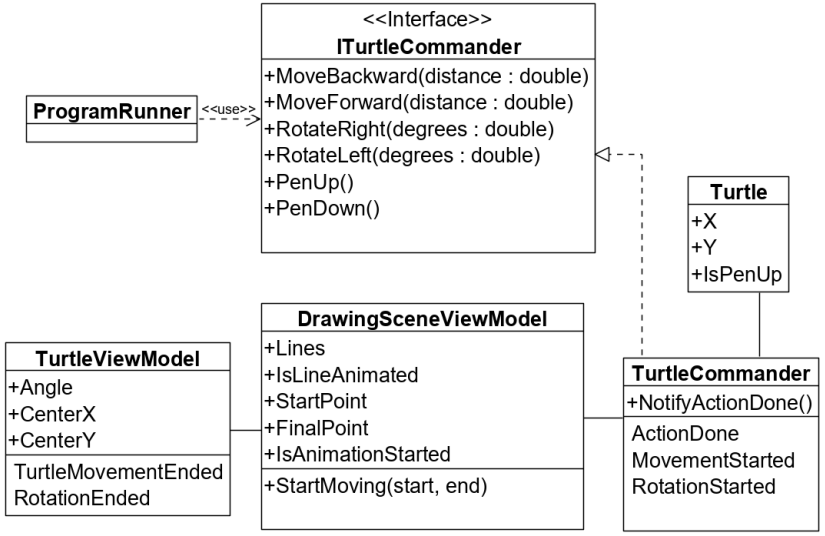
На указанном слайде изображено, как выглядит интерпретатор изнутри.
Есть несколько функций-парсеров, каждая из которых разбирает свой элемент метамодели. Эти функции комбинируются в одну, которая поочередно перебирает внутренние, пока не найдется подходящая.

Принцип работы интерпретатора на примере «Черепашки»



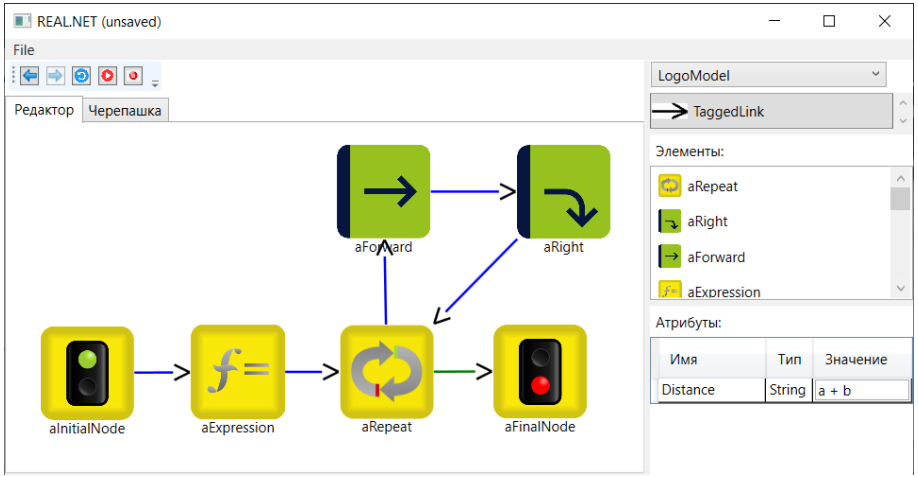
Здесь указано более подробно, как работает интерпретация на примере исполнителя «Черепашка». ProgramRunner получает команды, выделяемые интерпретатором при выполнении программы и вызывает соответствующие методы ITurtleCommander, управляющего «Черепашкой».
Интерпретатор «Робота» действует аналогично с той лишь разницей, что проверяет, что исполнитель не проехал через стенку.

Визуализатор на примере «Черепашки»



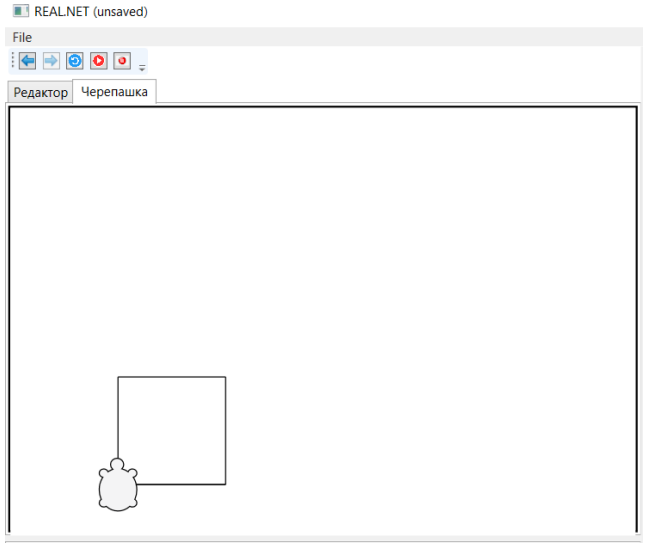
На данном слайде показана архитектура визуализатора также на примере «Черепашки». TurtleCommander уведомляет сцену о том, что нужно отрисовать какое-либо действие «Черепашки», она в свою очередь просит начать движение самого исполнителя. Когда тот закончит движение, он уведомит сцену об этом, та в свою очередь – TurtleCommander, а он обновит параметры «Черепашки».

Пример программы для «Черепашки»



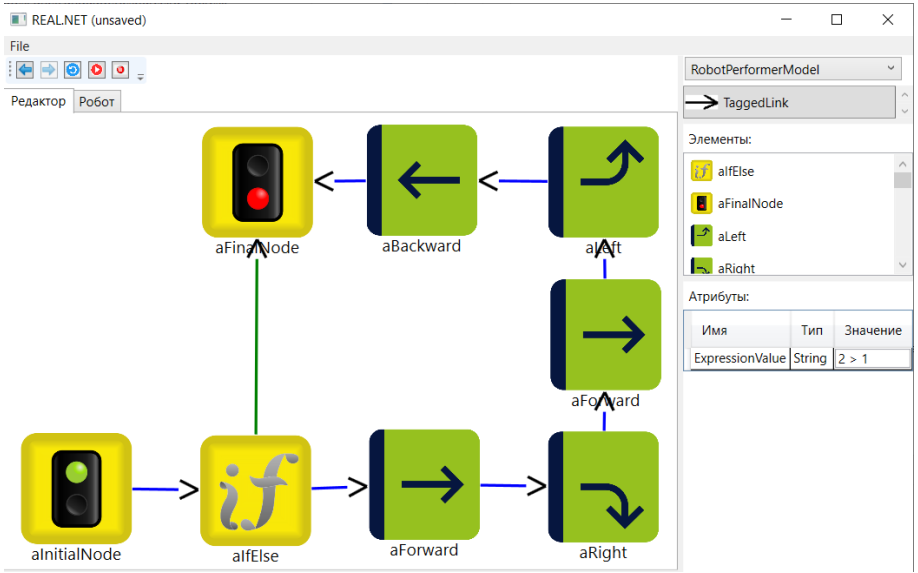
На данном слайде изображена программа для «Черепашки», рисующей квадрат. Справа внизу можно посмотреть атрибуты и их значения.

Визуализация программы



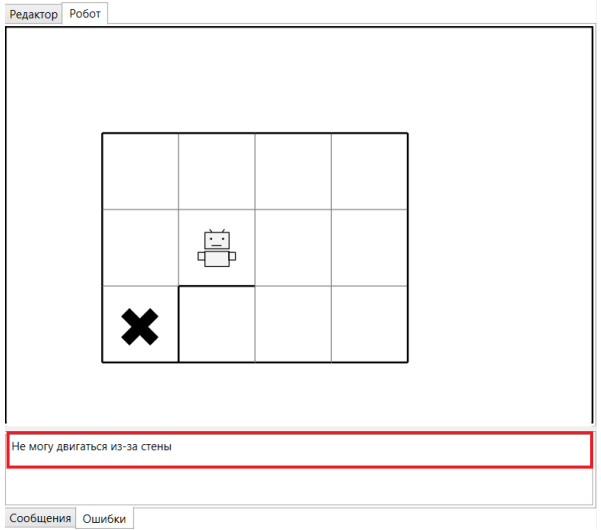
На данном слайде отображен результат выполнения программы, то есть нарисованный квадрат.

Пример программы для «Робота»



Здесь приведен пример программы для Робота.

Визуализация программы



На этом слайде отображена визуализация этой программы. Начинает «Робот» движение в клетке, отмеченной крестиком. Останавливается перед стенкой, и в консоли выдается ошибка, что он не может проехать.

Апробация

Первый этап

- Проведена апробация на учениках кружка 419 Лицея
- Были выданы задачи

Второй этап

- Проведен опрос 6 школьников после инструктажа и выполнения заданий
- Приложение набрало 75,4 балла по SUS, что соответствует оценке «хорошо» (B)

Апробация приложения была проведена в 2 этапа

1. Ученикам 419 Лицея была выдана инструкция, а также задачи. У детей были сложности с редактором, но через какое-то время они разобрались и сделали выданное задание.
2. Аналогичные задания и инструкции были выданы 6 знакомым школьникам. Был проведен опрос по шкале SUS(system usability scale), в результате приложение набрало в среднем 75,4 балла, что соответствует оценке «хорошо».

Результаты

- Сделан обзор существующих сред программирования исполнителей: FMSLogo, ЛогоМиры, КуМир
- Созданы визуальные языки для обоих исполнителей
- Разработана архитектура системы: спроектирован интерпретатор, а также механизм взаимодействия сервисных компонентов с интерпретатором
- Создан интерпретатор команд «Черепашки», его визуализатор был спроектирован и реализован
- Создан интерпретатор команд «Робота», был спроектирован и реализован его визуализатор
- Проведено тестирование полученных средств, а также апробация на реальных пользователях

Итого, полученные результаты [см на слайде].