Математические структуры и моделирование
2011, вып. 22, с. 74–85

УДК 004.588

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ СОСТАВЛЕНИЯ ОБУЧАЮЩИХ И ПРОВЕРОЧНЫХ ЗАДАНИЙ ПО ЕСТЕСТВЕННЫМ ЯЗЫКАМ

П. С. Долматова

In this article is presented special programming language for making natural language practice exercises and examinations. Special software for creating exercises and exams using this language has been developed. Examples of tasks using the developed language and software are considered.

Введение

В настоящее время при изучении языков широко используется компьютерное оборудование. Существуют различные программные комплексы для изучения языков и контроля знаний по языкам. Однако все подобные подходы были основаны на методиках изучения языка с помощью языка-посредника (чаще всего родного для человека языка), и не существовало средств и методов, основанных на независимом представлении понятий изучаемого языка и не использующих для обучения язык-посредник. Авторы [1, с. 20] предложили изучать языки посредством различных «действий» с предметами на экране без какого-либо другого языка-посредника. Это осуществляется следующим образом: слово и соответствующие действия демонстрируются пользователю; представляется то же самое слово и соответствующая начальная ситуация; пользователю нужно выполнить действие при помощи курсора; компьютер проверяет правильность действия, выполненного пользователем.

Если набор ситуаций фиксирован, то у пользователя может сложиться связь определённого понятия не только с изучаемым словом, но и с другими словами, присутствующими в соответствующих командах. Поэтому ситуации (задания) должны формироваться случайным образом, как предложено в [2, с. 217] и [4], т. е. при повторном запуске программы должны появляться другие, но аналогичные ситуации.

Таким путём могут быть продемонстрированы понятия, представляющие не только реальные объекты, но и воображаемые. В [6] демонстрируется естественное диалоговое представление абстрактных пространств.

Американский университет в Центральной Азии (г. Бишкек, Кыргызстан).
E-mail: dolmatova_p@mail.auca.kg

В [5] предложена общая схема программного обеспечения, которое включает в себя три составляющих: «Формализованное подмножество слов естественного языка», «Конструктор заданий» и «Интерфейс для обучения и тестирования» — необходимых для поставленных целей. Однако ранее это не было реализовано.

В [7] и [3] описана уже предложенная методика интерактивного компьютерного представления понятий естественных языков и её усовершенствованная реализация, позволяющая пользователю осуществлять более гибкое управление объектами на дисплее. И на этой основе построено эскизное программное средство для изучения киргизского языка без использования языков-посредников.

Далее может потребоваться изучение более обширных языковых понятий, которые более не могут быть выражены графически, поэтому возникает необходимость в ином способе представления обучающего материала. Материал можно представлять в виде текста на изучаемом языке. Если составлять набор заданий на каком-либо языке, то он будет конечным — обучаемый не сможет быть уверен, что получил правильное представление о том, что означают те или иные понятия. Следовательно, нужно каким-то образом составить задания так, чтобы их количество было бесконечным или хотя бы достаточно большим. Причём следует учесть тот факт, что в распоряжении составителя учебника зачастую бывают достаточно ограниченные ресурсы, например, время или ёмкость носителя информации, на котором будет распространяться обучающий материал. Выходом из такой ситуации может послужить случайное составление заданий с использованием генератора случайных чисел того компьютера, на котором обучаемый будет выполнять курс обучающих программ. Также нужно будет задать некий алгоритм генерации заданий, который позволит множеству генерируемых заданий оставаться большей частью осмысленным.

Все вышеизложенные требования были учтены при разработке программного обеспечения (ПО) для создания обучающих и контролирующих заданий по естественным языкам. Это программное обеспечение состоит из редактора заданий и основной программы — тестирующей или обучающей. Приложения имеют модульную структуру. Каждый модуль является обособленной частью набора типов заданий, которые требуются для того, чтобы полностью написать учебник или экзамен по языку. Данное ПО предназначено для специалистов-филологов, которые могут составлять с помощью неё вопросы для проверки знания того или иного языка человеком, отвечающим на вопросы. Поскольку далеко не все филологи владеют языками программирования, то для того, чтобы самостоятельно писать подобные программы, был разработан специальный язык, которому составитель экзамена может довольно быстро обучиться и, используя его, формировать задания. Разработанный язык называется TaskLang. Рассмотрим синтаксис этого языка.

1. Описание языка TaskLang

Программирование на языке TaskLang состоит из:

- определения фиксированных объектов;
- формирования случайных объектов;

- составления случайных заданий;
- формирования учебника или экзамена из заданий.

Всё это можно сделать в интерпретаторе языка TaskLang, который мы назвали Language Education System (LES). LES разработан на объектно-ориентированном языке C#. Для создания физических объектов использовались готовые библиотеки обработки столкновений Box2D, Catto, E. (2006–2007) и Box2DX, Kalasouski, I. (2008).

1.1. Объекты в TaskLang

Объекты бывают двух типов: фиксированные и случайные (множество однородных фиксированных объектов).

Фиксированный объект — это строка. Она может идентифицировать различные типы информации: текстовую, графическую и звуковую.

Если использован случайный объект, то вместо него из соответствующего множества подставляется (случайным образом) фиксированный объект. Если случайный объект используется в задании несколько раз, то он принимает одно и то же значение. Можно запросить второе значение этого же объекта, и тогда он принимает отличное от первого значение, но оно остаётся постоянным. Если задание выполнить второй раз, то значения случайных объектов изменятся в соответствии с этим правилом. Определение фиксированного объекта включает:

- имя объекта (object-name);
- класс, написанный на языке программирования C#, способный отвечать на внешние события и рисовать экземпляры данного класса (object-class). Такие классы могут быть взяты из стандартной библиотеки классов либо написаны пользователем на языке программирования C#;
- текстовое слово или список его грамматических форм (object-word);
- значения атрибутов. Стандартным атрибутом является object-word.

Синтаксис объявления фиксированных объектов:

[Declare*<object-name>#<object-class>|Word=
<object-word>|<attribute-name2>=<attribute-meaning2>|...]

Пример 1. Объявим два объекта:

[Declare*Rose#RoseDrawer|Word=роза|Color=розовый]

[Declare*Poppy#PoppyDrawer|Word=мак|Color=красный]

Определение случайного объекта включает:

- имя объекта;
- список фиксированных объектов.

Случайные объекты бывают двух типов:

- случайная последовательность фиксированных объектов;
- заданная последовательность фиксированных объектов.

Синтаксис объявления случайных объектов:

[<extended-object-name>~<primary-object-name1>|
<primary-object-name2>|...]

[<extended-object-name>=<primary-object-name1>|
<primary-object-name2>|...]

Ссылка на случайный объект:

[<extended-object-name>#<number>]

Она возвращает фиксированный объект по его номеру в определении случайного объекта.

Пример 2. Два случайных объекта:

[RandFlower~FlRose|FlPoppy]

[FixFlower=FlRose|FlPoppy]

[RandFlower#1] и [RandFlower#2] могут меняться, но всегда будут различными;

[FixFlower#2] всегда будет FlPoppy.

1.2. Составление случайного текстового задания

Формирование текста задания как последовательности

- строковых констант (включая слова, пробелы и знаки препинания);

  Синтаксис:

  <plain text>

- имён фиксированных объектов;

  Синтаксис:

  <plain text>

- имён случайных объектов (рассматриваемых в качестве ссылок на object-words) с определениями необходимых грамматических форм of object-words.

  Синтаксис:

  [GetProperty*[<extended-object-name1>#<number1>]|
  <attribute-name1>]

  или

  [GetProperty*<primary-object-name1>|<attribute-name1>]

Вследствие того, что в некоторых языках (например, киргизском, казахском) есть специальные алгоритмы словообразования, эти определения могут быть записаны как функции.

Синтаксис: [<language-code>*<word>|<primary-form-of-affix>].

Для других языков такие определения могут быть записаны как массивы: [Gram=word|grammar-form-1|grammar-form-2|grammar-form-3|...]

Пример 3. В этом примере мы хотим, чтобы на экране был один цветок. Этот цветок выбирается случайно. Им может оказаться роза или мак.

Для объявленного случайного объекта RandFlower, состоящего из FlRose со свойствами Word = роза, Color = розовый, и FlPoppy со свойствами Word = мак, Color = красный, мы вводим следующий текст задания:

Какого цвета [GetProperty*[RandFlower#1]|Word]?

И пользователю будет показан один из следующих вариантов:

Какого цвета мак?

Какого цвета роза?

1.3. Составление графического задания

- расположение случайных объектов в графическом окне;
- задание числа слоев для случайных объектов;
- задание направления медленного движения для некоторых случайных объектов.

Пример 4. (Продолжение Примера 3)

[Draw*[RandFlower#1]]

(число слоёв равно 1 и координаты нулевые (центр графической области) по умолчанию).

1.4. Составление словесного ответа в случайных заданиях

Это специальная функция, выполнение которой отображает приглашение для ввода текстового ответа во время выполнения задания. Приглашение для ввода отображается только тогда, когда в очереди событий все события, находящиеся ранее, уже выполнены.

Пример 5. (Продолжение Примеров 3 и 4)

[Answer*[GetProperty*[Flower#1]|Color]]

1.5. Составление ответа-действия к заданию

Ответ составляется как последовательность условий, применимых к действиям, совершаемым над изображением (во временном порядке). Одним из таких условий является утверждение о существовании или не существовании общих точек двух изображений объектов, рассматриваемых во время перемещения объекта

мышью. Также имеется возможность использовать отдельные события grasp и ungrasp для поднятия и отпускания объектов. Это последовательность логических выражений с текстовыми словами объектов и основными отношениями:

$$Intersect(object{-}name1, object{-}name2) \equiv (object{-}image1 \cap object{-}image2 \neq \emptyset);$$
$$Subset(object - name1, object - name2) \equiv (object - image1 \subset object - image2)$$

и операциями AND, OR, XOR, NOT. Эти последовательности определяют требуемую последовательность действий пользователя.

Если все условия выполнены, то на экран выводится ответ программы о том, что задание выполнено верно. Если не выполняется хотя бы одно из условий данной последовательности, то пользователю выдаётся сообщение о неверном ответе.

1.6. БНФ

Язык TaskLang определяется следующим образом:

ParseLanguage = TextCharacter, Brackets, TextCharacter

TextCharacter = (* Any UTF character except [ ] = | # * ~ / *)

Brackets = '[', Comment | FastChoise | KGZALG | MATH | DECLARE | GETPROPERTY | SETPROPERTY | DRAW | ANSWER | PLAYSOUND | DELAY | EVENT | INCORRECTEVENT | GRASP | UNGRASP | Extended Object | Assignment | RandomAssignment, ']'

Комментарий:

Comment = ParseLanguage

Быстрый выбор:

FastChoise = ParseLanguage, '|', ParseLanguage, '|', ParseLanguage

Алгоритм словообразования в киргизском языке:

KGZALG = 'KGZALG*', ParseLanguage, '|', ParseLanguage

Математические преобразования:

MATH = 'MATH*', NumericExpression

Объявление объектов:

DECLARE = 'DECLARE*', ParseLanguage, '#', ParseLanguage, '|', Parse Language, '=', ParseLanguage

Работа со свойствами объектов:

GETPROPERTY = 'GETPROPERTY*', ParseLanguage, '|', ParseLanguage

SETPROPERTY = 'SETPROPERTY*', ParseLanguage, '|', ParseLanguage, '=', ParseLanguage

Рисование:

DRAW = 'DRAW*', ParseLanguage

Ответ:

ANSWER = 'ANSWER*', ParseLanguage

Работа со звуком:

PLAYSOUND = 'PLAYSOUND*', ParseLanguage

Способ организации задержки:

DELAY = 'EVENT*', IntegerLiteral

Событие неверного ответа:
INCORRECTEVENT = 'INCORRECTEVENT*', BooleanExpression
Манипуляции с помощью курсора:
GRASP = 'GRASP*', ParseLanguage
UNGRASP = 'UNGRASP*', ParseLanguage
Случайный объект:
ExtendedObject = ParseLanguage, '#', IntegerLiteral
Фиксированное задание:
Assignment = ParseLanguage, '=', ParseLanguage
Случайное задание:
RandomAssignment = ParseLanguage, '~', ParseLanguage
Числовые выражения языка TaskLang:
NumericExpression = IntegerLiteral | ('-', NumericExpression) | (Numeric Expression, ('+' | '-' | '*' | '/'), NumericExpression)
IntegerLiteral = ('1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'), '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
Булевы функции:
BooleanExpression = ('NOT' BooleanExpression) | (BooleanExpression, ('or' | 'and'), BooleanExpression) | ('(', BooleanExpression, ')') | IntersectExpression | SubsetExpression
Пересечение:
IntersectExpression = 'INTERSECT(', ParseLanguage, ')'
Подмножество:
SubsetExpression = 'SUBSET(, ParseLanguage, ')'

2. Редактор модулей

Редактор модулей — это часть программы, предназначенная непосредственно для создания заданий на языке TaskLang.

Интерфейс редактора модулей (рис. 1) включает в себя два основных элемента для редактирования списка заданий и формирования готового набора заданий в виде дерева. После нажатия на кнопку «Добавить задание» создаётся задание выбранного типа и добавляется в список заданий. По списку можно перемещаться, нажимая кнопки «влево» и «вправо». Для того чтобы задание было показано пользователю, нужно, чтобы составитель включил это задание в дерево заданий. Сначала необходимо добавить раздел в дерево заданий, потом включить отображаемое задание как подпункт раздела с помощью контекстного меню. Одно и то же задание может быть добавлено в дерево заданий несколько раз. Так как задания включают в себя случайный элемент, то одно и то же задание будет воспринято пользователем как набор однотипных заданий.

В редакторе есть шаблоны типов заданий, с помощью которых составителю легче разрабатывать задания. Разработаны шаблоны для таких типов заданий, как диктант, текст, действия. Каждый шаблон реализован в виде отдельного модуля, поэтому в дальнейшем несложно будет расширить набор шаблонов,

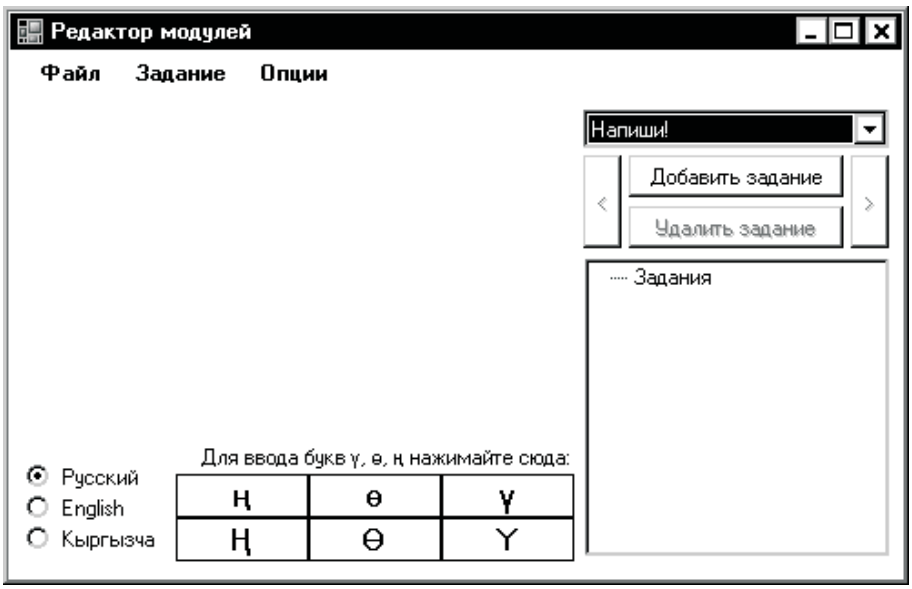

Рис. 1. Редактор модулей

просто добавив соответствующие модули. Рассмотрим интерфейс каждого из шаблонов в отдельности.

Задание типа диктант имеет своей особенностью то, что в нем обязательно присутствует звуковая составляющая. К этому типу можно отнести задания, где требуется написать произнесённое программой слово, ответить на аудиовопрос и др. В задание типа диктант можно включать звуковые файлы, текст, графические файлы. На рис. 2 представлен интерфейс редактора модулей с примером задания типа диктант. Здесь показаны следующие области:

1) область списка звуковых файлов. Если задание содержит один звуковой файл, то при каждом запуске этого задания будет воспроизведён звук из этого файла; если же задание содержит более одного звукового файла, то при запуске задания программа случайным образом выбирает и воспроизводит только один из звуковых файлов;

2) область графического изображения. Сюда при необходимости можно вставить графические файлы, соответствующие добавленным звуковым файлам;

3) область текста. В эту область при необходимости вставляется текст, обычно поясняющий основное задание, которое даётся при помощи соответствующего звука;

4) область кнопок используется для редактирования вариантов одного задания.

Задание типа текст в отличие от диктанта не содержат звуковых файлов, задание даётся с помощью текста. К такому типу относятся, например, задания: вставить пропущенные буквы, записать слово в требуемой форме (падеж,

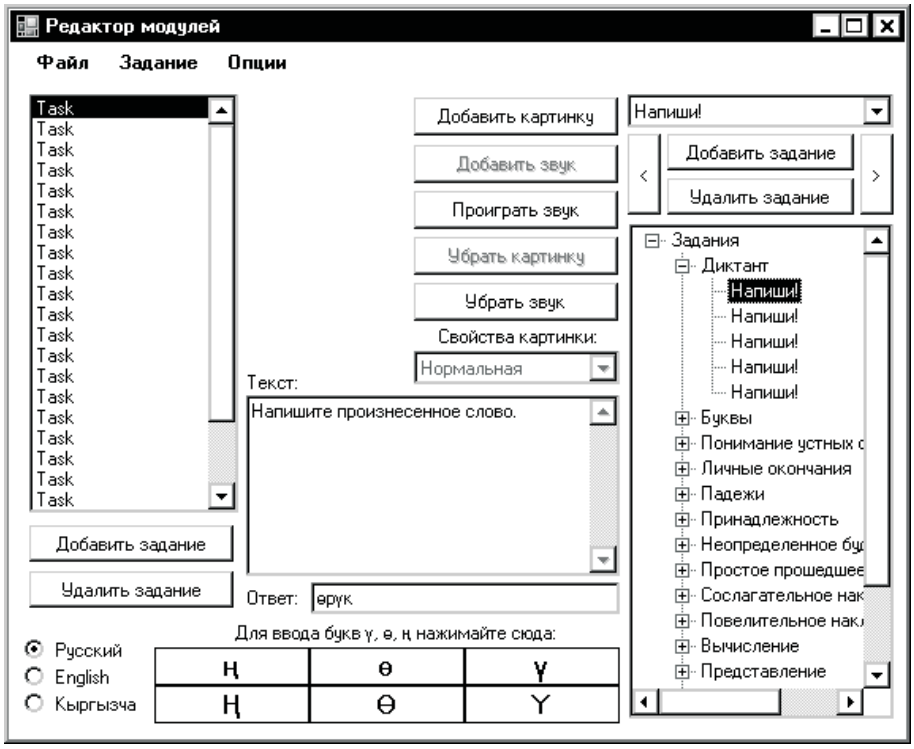

Рис. 2. Диктант

число, время и т. д.), понять текст и ответить на поставленные вопросы и др. Интерфейс редактора модулей с примером задания типа текст дан на рис. 3. Здесь показаны области:

1) область редактирования задания. В этой области на языке TaskLang составитель пишет задание. При составлении задания можно предусмотреть случайную генерацию задания;

2) область предварительного просмотра задания. В этой области отображается лишь тот текст задания, который будет выведен на экран в случае, если задание написано без ошибок, или будет выведено сообщение об ошибке, если таковая имеется. Эта область необходима для проверки правильности задания и его корректировки;

3) область предварительного просмотра начального значения строки ответа;

4) область отображения ответов, которые будут засчитаны программой как верные.

Задания типа действия пишутся на языке TaskLang. Эти задания требуют от пользователя выполнения некоторых действий с объектами на сцене с помощью курсора мыши.

Во вкладке Task definition (рис. 4) на языке TaskLang описывается сцена; тут же находится область предварительного просмотра задания. Во вкладке Scene

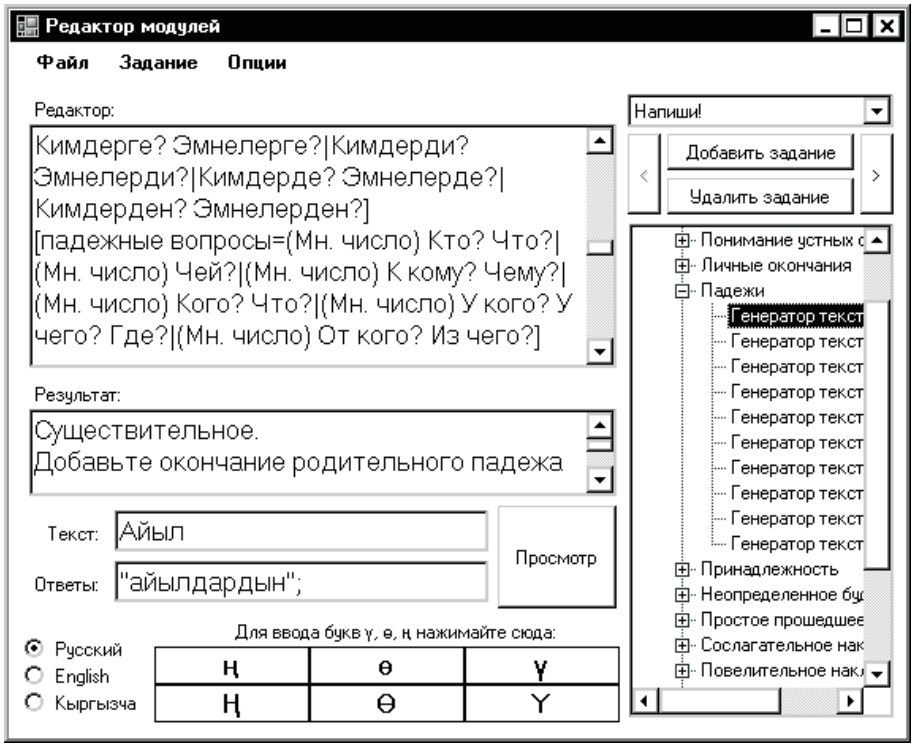

Рис. 3. Текст

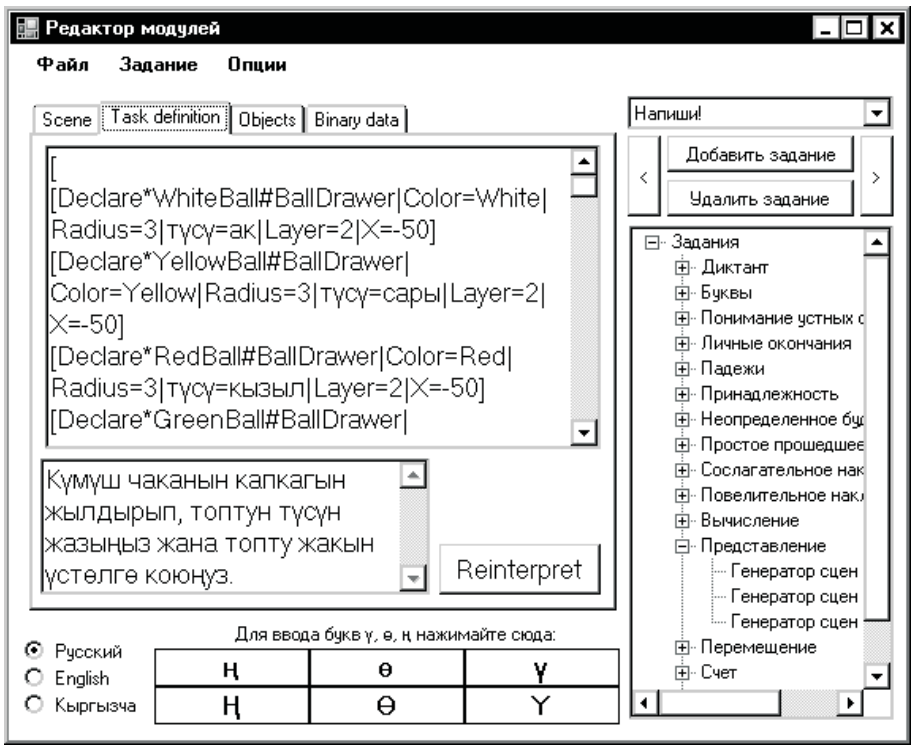

Рис. 4. Описание сцены на языке TaskLang

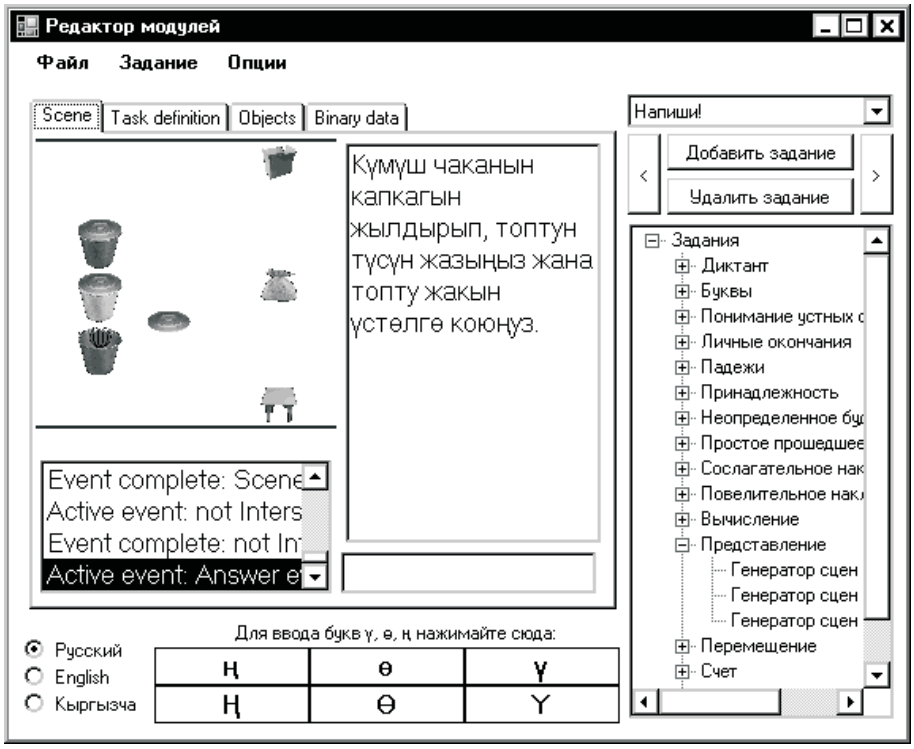

Рис. 5. Сцена

(рис. 5) изображается сцена. Рисунки можно вставлять из графических файлов и можно рисовать с помощью программы на языке TaskLang, в которой есть классы простейших геометрических фигур. Здесь же находится окно просмотра событий, в котором по очереди отображаются активные в данный момент события, а также сообщения об их успешном или не успешном окончании этих событий. Это окно предназначено для составителя заданий; здесь он может проверить правильность составленного им задания, отладить задание.

Вкладка Objects предназначена для создания новых объектов на языке C#. Во вкладке Binary data в задание можно добавлять звуковые и графические файлы.

Заключение

В заключение хотелось бы добавить, что с помощью программного обеспечения LES был создан первый вариант комплексного экзамена по киргизскому языку, некоторые задания из которого отображены на рис. 1–5. Этот экзамен был использован как демонстрационный в ряде вузов (Киргизско-Российский славянский университет, МУК, КГУСТА и др.), школ и в Институте теоретической и прикладной математики НАН КР. Экзамен выполняли как студенты и учащиеся, так и преподаватели, в том числе преподаватели киргизского языка. Участники экзамена дали положительную оценку ПО и разработанным с помощью него заданиям, проявили заинтересованность в использовании данного ПО в учебном процессе.

Литература

1. Панков П. С. Обучающая и контролирующая программа по словоизменению в кыргызском языке на ПЭВМ. Бишкек: Мектеп, 1992.
2. Панков П. С., Джаналиева Ж. Р. Опыт и перспективы использования комплекса UNIQTEST уникальных тестовых заданий в учебном процессе // Образование и наука в новом геополитическом пространстве: тез. докл. науч.-практ. конф. Бишкек: МУК, 1995.
3. Панков П. С., Долматова П. С. Построение дифференциальных уравнений для гибкого управления объектами с помощью компьютерной мыши // Исследования по интегро-дифференциальным уравнениям. Бишкек: Илим, 2007. Вып. 37. С. 18–23.
4. Pankov P. S. Independent learning for Open society // Collection of papers as results of seminars conducted within the frames of the program «High Education Support» / Foundation «Soros-Kyrgyzstan». Bishkek, 1996. Iss. 3. P. 27–38.
5. Pankov P. S., Alaeva S. A., Kutsenko V. A. Algorithmic Interactive Presentation of Notions // «Speech and Computer» SPECOM'2006: Proceedings of the 11th International Conference. Saint-Petersburg: Anatolya Publishers, 2006. P. 478–480.
6. Pankov P. S., Bayachorova B. J. Using computers to perform non-Euclidean topological spaces // The 6th conference and exhibition on computer graphics and visualization «Graphicon–96». Saint-Petersburg, 1996. Vol. 2. P. 73–84.
7. Pankov P. S., Dolmatova P. S. Algorithmical Language for Computer-Based Presentation of Notions // IKECCO'2007: Proceedings of the 4th International Conference on Computers and Techniques. Almaty, 2007. P. 274–279.