Инженерный вестник Дона, №3 (2014)
ivdon.ru/ru/magazine/archive/n3y2014/2522

Генератор уровней для игровых приложений

В.В. Селянкин, Н.А. Гуляев

Южный федеральный университет, Таганрог

Аннотация. Особенность большинства современных генераторов заключается в их узкой специализации, нацеленности на конкретный тип игровых уровней и приложений. Для разработки универсальных генераторов желательно предоставить пользователю возможность управления процессом генерации и различные дополнительные программные инструменты для обработки выходных данных, такие, как, например, инструменты конвертации геометрии преобразования текстур. Разработанный генератор реализован в виде динамически подключаемой библиотеки, являющейся ядром генератора, и набора функций обработки специфических данных и вспомогательных функций. Библиотека генератора реализует большинство функций для использования игровым приложением. Помимо библиотеки генератора, программный комплекс содержит редактор описательных структур и компоновщик баз элементов уровней. Предлагаемый генератор предназначен для разработки игровых программ.
Ключевые слова: игровые уровни, генератор игровых уровней, программный комплекс, управление процессом генерации, динамически подключаемая библиотека

Для всех игровых приложений, связанных с перемещением объектов сцены в пространстве, необходим такой ресурс, как игровой уровень, представляющий собой ту часть виртуальной реальности, где происходит перемещение игрока [1].Традиционный игровой уровень содержит набор статических объектов (основная геометрия уровня) и набор динамических (интерактивных) игровых объектов. Статические объекты представляют имитацию различных тел фиксированной конфигурации. При помощи статических объектов в игровом уровне могут моделироваться всевозможные твёрдые поверхности, рельеф местности, различные сооружения, составляющие некоторую сцену. Эти статические объекты служат некоторыми препятствиями, не позволяющими игроку покинуть пределы игрового пространства, либо, наоборот, могут способствовать перемещению в нём. Интерактивные игровые объекты могут представлять собой разнообразные сущности виртуального мира, управляемые пользователем, которые могут взаимодействовать между собой и с виртуальным миром. Это

могут быть движущиеся платформы, столь популярные в играх-платформерах, различные объекты инвентаря в играх-квестах, неигровые персонажи, или иные специфичные для конкретного игрового жанра объекты [2]. Игровые уровни традиционно проектируются человеком-дизайнером при помощи некоторых программных инструментальных средств [3]. Дизайнер, руководствуясь поставленным заданием и творческими намерениями, размещает геометрические элементы сцены–стены, постройки, препятствия и интерактивные игровые объекты неигровых персонажей, различные предметы и прочие специфичные для конкретной игры объекты [4].

Многие современные игровые приложения требуют обновления игровых уровней по содержанию, сложности и структуре для каждой игровой сессии [5]. В связи с этим возникает потребность в непрерывном получении и изменении игровых уровней. Кроме этого таким игровым приложениям требуются еще и уникальность каждого нового уровня, в результате чего возникает потребность в варьировании игровых уровней. Некоторые игровые приложения предусматривают возможность адаптации виртуального мира в зависимости от поведения игрока. Например, при необходимости корректировки режима сложности посредствам изменения всего игрового пространства, возникает потребность в некоторой интерактивности процесса создания и модификации существующих игровых уровней [6].

Таким образом, можно сформулировать три наиболее общих проблемы традиционного подхода к созданию уровней: требование непрерывности формирования сцен и уровней, требование возможности получения большого разнообразия уровней, а также требование интерактивности процесса. Решением данных проблем может служить автоматизация процесса создания игровых уровней, что является инновационным подходом, заключающимся в использовании генератора игровых уровней. В настоящее время существует

достаточно много разнообразных генераторов игровых уровней, однако, в силу тех или иных обстоятельств, в данном сегменте возникает несколько проблем, распространяющихся почти на все генераторы.

Особенность большинства современных генераторов заключается в их узкой специализации, нацеленности на конкретный тип игровых уровней или конкретные игровые приложения. В этой связи возникает проблема универсальности генераторов. Еще один нюанс заключается в слабой управляемости процессом генерации пользователем. В этом случае встаёт проблема управляемости процессом генерации. И, наконец, большинству генераторов требуются различные дополнительные программные инструменты для обработки выходных данных, такие, как, например, инструменты конвертации геометрии преобразования текстур. Таким образом, возникает еще одна проблема – независимость или автономность генератора. Подводя итог, можно сформулировать три общих проблемы данного инновационного подхода к созданию уровней: требование универсальности, требование управляемости и требование автономности.

Предлагается гибридный подход к созданию игровых уровней. Суть такого подхода заключается в частичной автоматизации процесса создания игровых уровней путём распределения ролей и степени участия человека-дизайнера и программы-генератора. Использование данного подхода позволит свести к минимуму, либо вовсе устранить, проблемы вышеописанных подходов, позволяя человеку решать проблемы универсальности, управляемости и автономности, а, программе – проблемы непрерывности, массовости и интерактивности. Программным воплощением данного подхода является автоматизированный генератор игровых уровней, описываемый далее.

Основной целью данного решения является удовлетворение потребностей в игровых уровнях, как разработчиков игровых приложений,

так и конечных пользователей. На сегодняшний день существуют методы получения уровней, основывающиеся на концепции распределения ролей [7–11]. Однако большинство таких разработок имеет незавершённый статус, либо узкую специализацию, либо вовсе являются закрытыми коммерческими продуктами [12–14]. Кроме того, целью данного программного решения является реализация принципа частичной автоматизации создания игровых уровней для широкого круга разработчиков игровых приложений.

Такой подход решения задачи может использоваться как на этапе разработки игрового приложения, так и на этапе его эксплуатации. В этом случае на этапе разработки генератор позволяет получать игровые уровни с последующим включением их в ресурсную часть игрового приложения, а также выполнять мануальный отбор сгенерированных уровней по каким-либо критериям. Кроме этого разработчик-дизайнер игрового приложения может выполнять корректировку исходных данных генератора для получения желаемого результата. Круг пользователей в данном случае сводится к разработчикам и дизайнерам, участвующим в разработке игрового приложения. На этапе эксплуатации генератор используется игровым приложением для получения новых игровых уровней на основе сформированных разработчиками и дизайнерами исходных данных. Конечный пользователь может задавать некоторые управляющие параметры, либо они могут быть сформированы в процессе использования игрового приложения.

Предлагаемое решение подразумевает распределение обязанностей между человеком и программой следующим образом. Разработчиком-дизайнером формируются исходные данные – некоторые элементы будущего уровня и управляющие значения для генераторов числовых последовательностей, а программный модуль производит построение различных комбинаций этих элементов на основе управляющих значений.

Таким образом, человек имеет полный контроль над процессом генерации, формируя по собственному усмотрению различные наборы элементов уровней и задавая управляющие значения, иными словами – входные данные полностью формируются человеком. Программный модуль, в свою очередь, ограничен только лишь формированием различных сочетаний из заданного множества элементов уровня в соответствии с управляющими значениями.

В данном случае дизайнер имеет возможность проектировать игровой уровень почти так же, как и при традиционном подходе. В отличие от него вместо монолитного игрового уровня требуется создать некоторое число отдельных элементов уровня, логически связанных между собой, и поместить их в базу элементов уровня, которая будет передана модулю генератора (рис. 1).

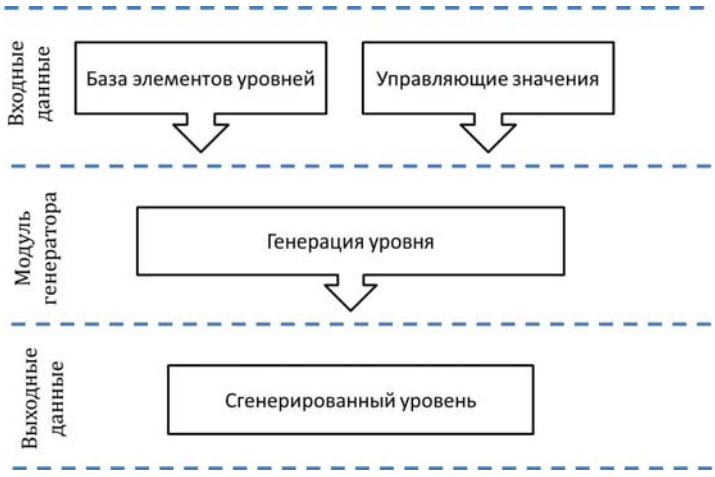

Рис. 1. – Процесс создания уровня

База элементов уровня представляет собой набор специальных описательных структур, несущих в себе основные параметры элемента уровня – координаты в пространстве, поворот, размер. База элементов может содержать и реальные трёхмерные модели элементов уровня, которые будут впоследствии размещены в игровом пространстве, будучи привязанными к конкретной описательной структуре. Однако база может ограничиваться лишь описательными структурами, предоставляя игровому приложению возможность самостоятельно производить построение геометрии.

Описательные структуры создаются человеком в специальном редакторе, поставляемом с данным генератором, после чего все необходимые описательные структуры помещаются в базу элементов уровня и далее используются непосредственно модулем генератора. Сам генератор ориентирован на тайловые уровни. Это означает, что минимальной единицей представления элемента уровня является некоторый блок (тайл) фиксированного размера и, возможно, фиксированной ориентации. Похожее представление объектов используется при решении аналогичных задач, например, компоновки трехмерных геометрических объектов [15]. Таким образом, каждый элемент уровня должен представлять собой либо один вполне самостоятельный тайл, либо состоять из некоторого числа тайлов. Таким образом, описательные структуры представляют собой описание некоторого тайла, либо группы тайлов [16]. Использование тайлов позволяет в значительной степени сократить вычислительные затраты, связанные с различными операциями над геометрическими объектами в пространстве. Кроме того, при помощи тайлов классификация участков уровня по проходимости становится достаточно простой, позволяя избегать появления непроходимых участков или тупиков.

Выходные данные модуля генератора имеют описательный вид, фактически, результатом работы генератора является некоторое количество описательных структур, расположенных в пространстве и согласованных друг с другом некоторым образом. Описательное представление уровня позволяет рассматривать каждый элемент уровня как предписание по расположению, повороту и размеру для некоторого реального элемента уровня, представленного трёхмерной моделью, спрайтом и т.п. Любое количество описательных структур, согласованных друг с другом может считаться завершённым игровым уровнем в описательном представлении. Таким образом, получив уровень в описательном представлении, игровое

приложение сможет на основании полученных предписаний произвести либо построение геометрии, либо размещение заранее подготовленных элементов из базы (рис. 2).

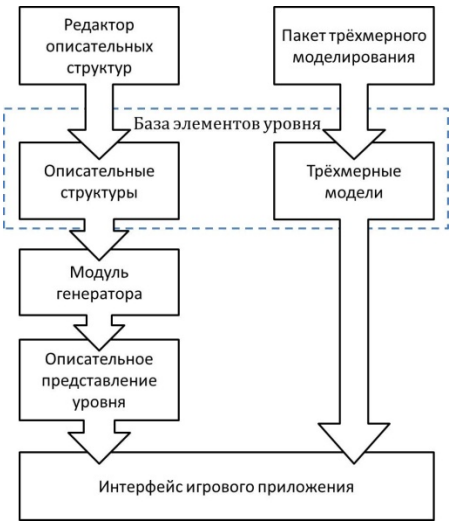

Рис. 2. – Этапы обработки данных

Однако, игровому приложению необходим специальный модуль, который был бы способен распознать и вычленить информацию из переданного ему уровня в описательном представлении, а также, при необходимости, инициировать процесс генерации нового игрового уровня. Такой модуль представляет собой некоторый интерфейс между библиотекой генератора и игровым приложением. Интерфейс игрового приложения может являться частью игрового приложения, либо быть самостоятельным модулем. Такой интерфейс не входит в состав генератора, кроме того, каждому игровому приложению необходим собственный интерфейс, который учитывал бы особенности конкретного игрового приложения. Разработка интерфейса игрового приложения – обязанность разработчика игрового приложения, которое должно использовать генератор. Разработка, либо подбор уже существующего интерфейса – единственное, что требуется от разработчика для выполнения привязки генератора к игровому приложению.

Ядро модуля генератора разделено на три подмодуля, каждый из которых играет свою роль в процедуре генерации уровня. Разделение ядра

подобным образом позволяет разбить весь процесс генерации на отдельные этапы, в то же время, позволяя производить многопроходную обработку и обработку различными подмодулями одного класса.

Модуль генерации маршрута определяет изначальные направления и расположения для элементов уровня, представленными описательными структурами. Генерацию маршрута, возможно, производить по различным алгоритмам – от построения простого пути до выстраивания сложных траекторий с ветвлением. Входными данными генератору маршрута передаются управляющие значения, регулирующие большинство аспектов выбранного алгоритма. Результатом работы данного модуля будет являться набор описательных структур, несущих данные о расположении, повороте и размере каждого тайла. Такой набор может быть передан следующему подмодулю по принципу конвейера, либо снова передан какому-либо модулю, выполняющему генерацию маршрута для повторной обработки (рис. 3).

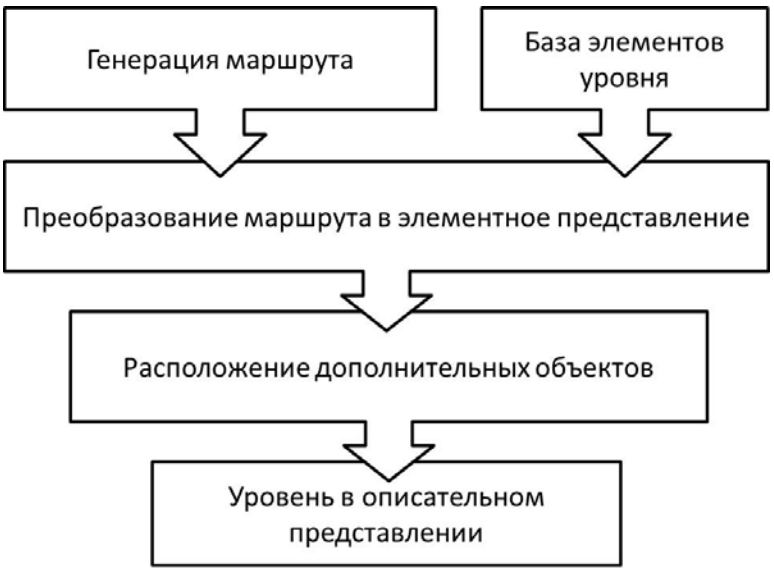

Рис. 3. – Этапы генерации уровня

Многопроходная обработка в данном случае может быть полезна при построении структурно сложных уровней, требующих участия сразу нескольких типов генераторов маршрута. Модуль преобразования маршрута

в элементы уровня позволяет классифицировать группы тайлов ранее сгенерированного маршрута, преобразуя маршрут из тайлового представления в элементное представление. Фактически, обработка уровня на данном этапе заключается в замене групп тайлов сгенерированного маршрута группами тайлов из базы элементов уровня. Данный модуль на входе получает некоторый сгенерированный маршрут и базу элементов уровня, сформированную пользователем, а на выходе предоставляет массив описательных структур, несущих соответствующие данные уже элементов уровней. Третий модуль выполняет расположение дополнительных объектов (интерактивных элементов уровня) на сгенерированном уровне. Данный модуль на вход получает уровень в элементном представлении, а на выходе предоставляет этот же массив, дополненный описательными структурами, несущими данные о расположении дополнительных объектов. Однако данный модуль не может предусмотреть все особенности интерактивных объектов какого-то конкретного игрового приложения, такие как модели поведения неигровых персонажей, количество здоровья игрока или количество объектов инвентаря. Поэтому результат работы данного модуля должен, при необходимости, быть дополнен, либо скорректирован интерфейсом игрового приложения, принявшим сгенерированный уровень. В итоге генерируемый уровень, пройдя достаточное количество итераций обработки каждым модулем, будет выдан в качестве выходных данных генератора.

Генератор реализован в виде динамически подключаемой библиотеки, заключающей в себе непосредственно ядро генератора, функции обработки специфических данных, таких как описательные структуры, а также некоторые вспомогательные функции, такие как сохранение результата генерации в файл и загрузка сгенерированного уровня из файла. Библиотека генератора предоставляет большинство функций для использования

непосредственно игровым приложением, а точнее – интерфейсом игрового приложения. Помимо библиотеки генератора, программный комплекс содержит редактор описательных структур и компоновщик баз элементов уровней, данные программные средства предназначены для использования разработчиками (дизайнерами уровней) на этапе разработки.

Программный комплекс состоит из библиотеки генератора и редактора. К программному комплексу прилагается готовый интерфейс игрового приложения, предназначенный для движка Unreal Engine. Библиотека генератора состоит из нескольких модулей, каждый из которых решает конкретную задачу. В каждом модуле определены специализированные классы, решающие часть конкретной задачи, решению которой посвящён их модуль. В модуле, производящем генерацию маршрута, определены классы, производящие генерацию маршрута по различным алгоритмам. Эти классы выстраиваются в иерархию от класса, реализующего самый простой алгоритм к классам, реализующим более сложные алгоритмы. В модуле, производящем преобразование маршрута из тайлового представления в элементное представление определены специальные классы, представляющие шаблоны и элементы уровня. В модуле, производящем комплексную генерацию, определен специальный класс, включающий в себя классы, производящие генерацию маршрутов, преобразование уровня в элементное представление, а также некоторые вспомогательные классы. Данный класс обеспечивает комплексный подход, производя и генерацию маршрута, и генерацию геометрических объектов. Более того, использование такого класса позволяет обеспечить конвейеризацию процесса и целостность промежуточных данных. Взаимодействие с классами генераторов предполагается только через данный класс. Для корректной работы библиотеки генератора требуется операционная система семейства Windows, (Windows XP и выше). Необходимое количество свободной оперативной

памяти зависит от размеров базы шаблонов и размера генерируемого уровня. Для возможности работы в редакторе пользователю необходимы базовые устройства ввода/вывода, а также доступ к файловой системе для возможности сохранения и загрузки баз шаблонов. Количество необходимого свободного места на диске определяется размером самих баз шаблонов.

В настоящее время программный комплекс позволяет производить генерацию нескольких типов уровней для двухмерных и трёхмерных игровых приложений. Степень детализации устанавливается пользователем. Уровень может состоять как из элементов, размеры которых многократно превосходят размеры игровых персонажей, так и, наоборот, из элементов намного меньших. И здесь возможно даже сведение генерации уровня к построению одного или нескольких воксельных объектов, либо поверхности [17]. В программный комплекс включён интерфейс для игровых приложений, использующих игровой движок Unreal Engine 3. Таким образом, большинство игровых приложений на данном игровом движке смогут использовать готовый интерфейс игрового приложения, что существенно облегчит задачу их разработчиков. Редактор описательных структур на данный момент имеет достаточный функционал, позволяющий дизайнеру игровых уровней создавать описательные структуры и формировать базы элементов уровня.

Основной акцент при разработке данного программного комплекса сделан на разнообразие типов генерируемых уровней. Соответственно, в библиотеку генератора планируется добавление большего количества модулей генераторов маршрутов, и, возможно, модулей расположения интерактивных объектов. Данный программный комплекс с открытым исходным кодом позволяет различным разработчикам дополнять и модифицировать его. Благодаря своей универсальности и доступности разработка может стать некоторой основой для других разработок в данной

области или смежных областях. Возможно использование системы в качестве вспомогательного программного обеспечения для различных прикладных задач. Например, для системы, выполняющей визуализацию результатов моделирования геодезических расчётов, как например [18], возможно построение внешней среды (построек и различных сооружений) при помощи данного генератора и другие.

Литература

1. McGuire M., Jenkins O. C. Creating Games: Mechanics, Content and Technology. A.K. Peters Ltd., USA, 2008. pp. 267-269.
2. Byrne E. Game Level Design. Charles River Media, USA, 2004. p. 273.
3. Feil J.H., Scattergood M. Beginning Game Level Design. Thomson Course Technology, USA, 2005. pp. 9, 11, 13.
4. Cross N. Design Thinking: Understanding How Designers Think and Work. Berg Publishers, UK, 2011. pp. 28.
5. Steinkuehler C. Massively multiplayer online gaming as a constellation of literacy practices // The Design and Use of Simulation Computer Games in Education. The Netherlands: Sense Publishers, Netherlands, 2007. pp. 187-212.
6. Hunicke R., Chapman V. AI for Dynamic Difficulty Adjustment in Games // 2004 AAAI Workshop. The AAAI Press, USA, 2004. pp. 91-96.
7. Smith G., Whitehead J., Mateas M. Tanagra: A mixed-initiative level design tool // FDG ’10: Proceedings of the Fifth International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 3-5, 7-8.
8. Smelik R., Tutenel T. Integrating procedural generation and manual editing of virtual worlds // FDG ’10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 3-6.
9. Hastings E.J., Stanley O.K. Interactive genetic engineering of evolved video game content // FDG ’10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 2-4.

10. Whitehead J. Toward procedural decorative ornamentation in games // FDG '10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 1-4.

11. Ashmore C. Key and Lock Puzzles in Procedural Gameplay // School of Literature, Media, and Communication Masters Projects. Georgia Institute of Technology, USA, 2007. pp. 22-26.

12. OBLIGE Level Maker [electronic resource] // sourceforge.net – URL: oblige.sourceforge.net (Accessed: 08.02.2014)

13. Angdoom [electronic resource] // interreality.orgt – URL: interreality.org/~tetron/technology/angdoom/ (Accessed: 08.02.2014)

14. SLIGE [electronic resource] // sourceforge.net – URL: doomworld.com/slige/ (Accessed: 08.02.2014)

15. Ягудин Р.Р. Оптимизация компоновки трехмерных геометрических объектов на основе годографа вектор-функции плотного размещения // Инженерный вестник Дона, 2012, № 3 URL: ivdon.ru/ru/magazine/archive/n3y2012/921.

16. S. Bjork S., Holopainen J. Patterns in game design. Charles River Media, 2004, USA. pp. 112, 150-155.

17. Бланко Л.М.Л., Березняк С.А., Пантелюк П.А. Алгоритм компрессии воксельного поля, оптимизированный для хранения данных о ландшафте (ANIRLE) // Инженерный вестник Дона, 2013, № 4 URL: ivdon.ru/magazine/archive/n4y2013/1997.

18. Лахов А.Я. Программное обеспечение для стереовизуализации результатов конечно-элементного моделирования // Инженерный вестник Дона, 2013, № 1 URL: ivdon.ru/magazine/archive/n1y2013/1501

References

1. McGuire M., Jenkins O. C. Creating Games: Mechanics, Content and Technology. A.K. Peters Ltd., USA, 2008. pp. 267-269.

2. Byrne E. Game Level Design. Charles River Media, USA, 2004. p. 273.

3. Feil J.H., Scattergood M. Beginning Game Level Design. Thomson Course Technology, USA, 2005. pp. 9, 11, 13.

4. Cross N. Design Thinking: Understanding How Designers Think and Work. Berg Publishers, UK, 2011. pp. 28.

5. Steinkuehler C. Massively multiplayer online gaming as a constellation of literacy practices // The Design and Use of Simulation Computer Games in Education. The Netherlands: Sense Publishers, Netherlands, 2007. pp. 187-212.

6. Hunicke R., Chapman V. AI for Dynamic Difficulty Adjustment in Games // 2004 AAAI Workshop. The AAAI Press, USA, 2004. pp. 91-96.

7. Smith G., Whitehead J., Mateas M. Tanagra: A mixed-initiative level design tool // FDG ’10: Proceedings of the Fifth International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 3-5, 7-8.

8. Smelik R., Tutenel T. Integrating procedural generation and manual editing of virtual worlds // FDG ’10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 3-6.

9. Hastings E.J., Stanley O.K. Interactive genetic engineering of evolved video game content // FDG ’10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 2-4.

10. Whitehead J. Toward procedural decorative ornamentation in games // FDG ’10 International Conference on the Foundations of Digital Games. ACM New York, USA, 2010. pp. 1-4.

11. Ashmore C. Key and Lock Puzzles in Procedural Gameplay // School of Literature, Media, and Communication Masters Projects. Georgia Institute of Technology, USA, 2007. pp. 22-26.

12. OBLIGE Level Maker [electronic resource] // sourceforge.net – URL: oblige.sourceforge.net (Accessed: 08.02.2014)

13. Angdoom [electronic resource] // interreality.orgt – URL: interreality.org/~tetron/technology/angdoom/ (Accessed: 08.02.2014)

14. SLIGE [electronic resource] // sourceforge.net – URL: doomworld.com/slige/ (Accessed: 08.02.2014)

15. Jagudin R.R. Inženernyj vestnik Dona (Rus), 2012, № 3 URL: ivdon.ru/ru/magazine/archive/n3y2012/921.

16. S. Bjork S., Holopainen J. Patterns in game design. Charles River Media, 2004, USA. pp. 112, 150-155.

17. Blanko L.M.L., Berezhnjak S.A., Panteljuk P.A. Inženernyj vestnik Dona (Rus), 2013, № 4 URL: ivdon.ru/magazine/archive/n4y2013/1997.

18. Lahov A.Ja. Inženernyj vestnik Dona (Rus), 2013, № 1 URL: ivdon.ru/magazine/archive/n1y2013/1501