**Semi-Automated Game Generation Module**

**Summer 2013**

**Tech Report**

**Kaleb Breault**

**Chris Mojica**

**Abstract:**

The use of serious educational games has many advantages, offering immersive, engaging and fun environments that require deep thinking and complex problem solving within a construct of overcoming obstacles and challenges. Developing new games, however, to support broad and rapidly evolving disciplines has remained time consuming, expensive, and requiring the expertise of game designers, software developers, software engineering educators, and players. Here, we introduce a semi-automated component-based engineering approach for generating serious educational games, which enables educators to rapidly and independently develop their own games across diverse educational topics. An update on the progress of this approach is explained in detail in this report [11].

**Table of Contents**

**Abstract**

**Table of Contents**

**1. Introduction 1**

**2. Methodology 2**

**2.1 Interactive Wizard UI 2**

**Fig. 1** Overview **3**

**2.2 Create a SEGCB 2**

**2.3 Repository 4**

**3. Background 5**

**Fig. 2** Original AHP Work **6**

**Fig. 3** Additional Work **6**

**4. Game Components 8**

**Fig. 4** Component Dependencies Overview **8**

**4.1 Character Layer 8**

**Fig. 5** Character Component Overview **8**

**4.2 Lesson Layer 9**

**Fig. 6** Challenge Component Overview **9**

**4.3 Challenge Layer 10**

**Fig. 7** Challenge Component Overview **10**

**4.4 Locale Layer 11**

**Fig. 8** Locale Component Overview **11**

**4.5 Subject Layer 12**

**Fig. 9** Subject Component Overview **12**

**4.6 Theme Layer 12**

**Fig. 10** Theme Component Overview **12**

**Fig. 11** IntroScreens Overview **13**

**Fig. 12** OutroScreen Overview **14**

**Fig. 13** ThemeStories Overview **14**

**4.7 Selection Process 14**

**5. Validation 15**

**Fig. 14** Test Cast #1 **15**

**Fig. 15** Test Case #2 **17**

**6. Conclusion and Future Work 19**

**7. References 20**

**1. Introduction**

Educational infrastructures face significant challenges including the need to rapidly, widely, and cost effectively introduce new or revised course material; encourage the broad participation of students; support traditional and emerging “classrooms” (e-learning, distributed classrooms), and address changing student motivations and attitudes. The course material needs to address learning objectives, which span subject specific content and transferable skills, such as collaboration, critical thinking, creative thinking, problem solving, reasoning abilities, learning to learn, professionalism, and decision making [11].

Serious educational games (SEGs) have significant pedagogical potential as they provide immersive, engaging and fun environments that require deep thinking and complex problem solving within a construct of overcoming obstacles and challenges [1][3][8][10]. They create interactive student-centered environments rather than a passive content-centered classroom environment. This allows students to create a personalized learning experience, progressively incorporating new knowledge and scaffolding it into what they already know. Because each student is able to engage course-based material at his or her own pace; underprepared or at-risk students can focus on needed skills at their convenience. Feedback is frequent and immediate, thereby reinforcing mastery of fundamental skills required for advancing further into the game. Integrating games into curriculum creates a highly motivating learning environment; it draws on students’ sense of fantasy and amusement; it is self-directed, appealing to individual student’s curiosity; and it is a continuous challenge wherein any existing tasks or knowledge that appears incomplete, inconsistent or incorrect motivates a student to foster deeper levels of learning [11].

There are many potential benefits to adopting games into curricula, but where do we find these games? As with other software applications, we can consider acquiring them off-the-shelf (e.g.,[6]), modifying an existing game (e.g., [5][7]), or developing new ones. Games are complex applications; developing new ones has been time consuming, expensive, and has required substantial expertise from diverse stakeholders: game developers, software developers, educators, and players. Research in the game community to improve this situation with semi-automated game generation approaches is just beginning to receive attention (refer to the related work discussion in Section IV).

The component-based software engineering community continues to invest substantial effort to support the timely, cost effective development of large-scale, complex systems [2][9]. At a very high level, this community considers software development as a problem involving the selection and composition of re-usable, high quality components, like assembling “lego” blocks. The components have gone through a resource intensive development process to specify, design, implement, test, and document; they have comprehensive interface descriptions including a specification of their functional and non-functional capabilities. The non-functional description includes quality of service attributes (e.g., performance). The specification, selection, and composition of components have received considerable attention. For example, formal notations, which offer a means to specify components concisely and unambiguously, such as XML, fuzzy logic, first-order logic as well as architectural description languages and coordination languages have been explored [11].

To the best of our knowledge, a component-based software engineering approach for rapidly developing serious educational games across diverse educational topics is not available in the literature. Here, we introduce a new semi-automated component-based serious educational game generation approach, SEGCB, and present our preliminary results on the game component specification using a small example game to illustrate. Six game components are defined in the SEGCB (theme, locale, subject, characters, lesson, and challenge), which cut across the game. The components are viewed as layers of a game; they have dependencies in order to generate a consistent game. For example, characters may appear throughout the game: across the game introduction, in a challenge midway through, and the wrap-up. The game components are specified in XML, a standard, formal notation that is considered straightforward to understand and load in tool support with established libraries. The components are assembled into games using and output as game specifications in XML. The games generated are in the SimSYS [4] format [11].

**2. Methodology**

The SEGCB approach is organized into three layers (Fig. 1): interactive wizard interface; create components based serious educational game engine; and the repository [11].

## 2.1 Interactive Wizard User Interface

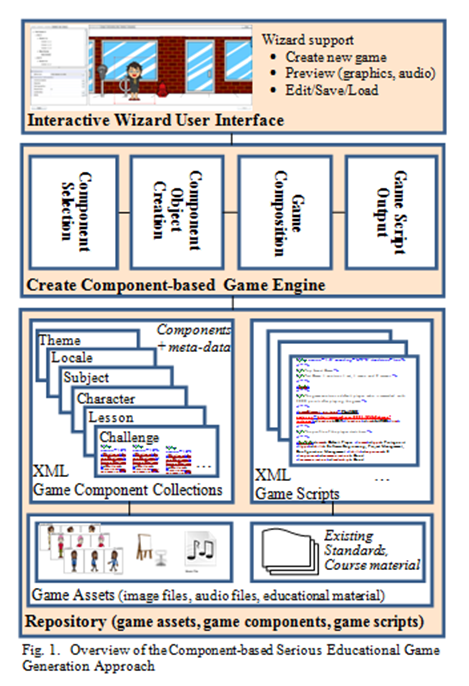
The User Interface provides wizards to acquire information from the game developer; preview the game generated; and edit the game generated as needed. The information acquired from the game developer is used to characterize and create a useful game. The information includes: game subject area; target player audience; intended context to use the game; level of difficulty; length; and the style of the game. For example, a teacher for grade 4 students in WI, U.S.A. may need a short game to use in his class as an early part of the lesson. The teacher may characterize the game needed as follows:

· subject area: basic algebra

· target player: grade 4 student in WI, U.S.A.

· use of the game: in class time

· level of difficulty of the game: easy

· length of the game: short

· style of the game: cartoon

· educational standard: WI state standards for grade 4 mathematics

The wizard provides the options the game developer can select from; the current prototype uses dropdown menus [11].

For example, the user can choose from subject areas (basic algebra, reading), levels of difficulty (easy, moderate, difficult), length of game (short, moderate long), target player (grade or experience level). As additional repositories of subject area assets are added to the repository, the options can be readily updated. In the future it will be interesting to explore alternative interface techniques, natural language processing techniques for speech or text input.

Using the information acquired, the game developer can create the game; the Create Component-based Serious Educational Game Engine is used to accomplish this. The teacher can preview the generated game, edit it if desired, and save the game [11].

## 2.2 Create Component-based Serious Educational Games

The Create Component-based Serious Educational Game Engine is a pipeline of four modules: component selection; component object creation; game composition; and game script output. Currently, the search, ranking, and selection are done manually for the component selection; the object creation, composition, and game script output modules have been prototyped [11].

The Component Selection module uses the characterization information provided by the game developer to search, rank, and select a collection of game components that are used to assemble a game. The game components are retained in the Repository; there are collections of theme, locale, subject, character, lesson, and challenge components specified in XML. Meta-data about the collections are used in the search; these are also represented in XML. The output of this module is a collection of the selected game components’ names (e.g., file names); this collection is used by the Component Creation module [11].

The Component Creation module uses the collection of component names provided to it retrieve the components from the Repository and convert them into a collection of working objects. The component creation is in a separate module in order to encapsulate the capability to load the XML files from the Repository. The output of this module is a collection of working objects; this collection is used by the Game Composition module.

The Game Composition module assembles a game out of the components by using an overall game structure as a wireframe. The games are organized as a sequence of Acts (intro act, internal acts, outro act); the acts have screens. For example, an internal (learning) act has screens to introduce, present, and wrap-up lessons and challenges. The acts are built in order: Intro Act, Learning Act(s), then the Outre Act. Once all the acts are created, they are added to the game’s wireframe [11].

Once the game is built it can be passed to the game script output module, which exports the game into the desired format. Currently, the game is output as an XML file in the SimSYS game specification format [11].

## 2.3 Repository

The repository contains the re-usable game components, game assets, and game scripts. The game components, described in more detail in Game Components, are: challenge, characters, lesson, locale, subject, and theme. The challenge entity represents the challenge presented at the end of the lesson and is currently limited to a multiple choice quiz of varying length. The characters entity contains information about what names and assets should be used for each of the four supported character types (hero, villain, player, alternate). The lesson entity encompasses a single lesson which directs the student to learn some objective. The locale entity holds all information describing the location of the experience such as the background, character positions, background and foreground object positions and locations of where text can be displayed. This allows other entities to simply reference a named location, such as a character’s speech bubble, and provide the accompanying text. The subject entity describes the overall subject that is being taught and any introductory text. The theme is the story aspect of the game and contains not only the information used to build the intro and outro acts, but also story snippets, which surround each learning act to progress the story throughout the learning experience [11].

The game asset collection in the repository includes graphic images, audio files, and existing standards and course material. The graphic images are organized by characters, props, backdrops, and interactions; images are in the standard .png format, 200x300 pixels. For each character, there are 56 images to provide a variety of poses (standing, sitting, walking, talking, facing different directions (left, right, straight ahead). Images of props include furniture, easels, podiums, clocks, computers, phones, and so on. The backdrops provide a background image for a setting. For example, backdrop images for offices, classrooms, meeting rooms, medieval castles, forests, outer space, and so on are stored for re-use. The interaction images include information boxes, conversation bubbles, and buttons. Audio files include sound effects and music; these are stored in standard .wav or .mid files format. Existing standards and course material are also stored in the repository as a valuable reference material. Course material may be in presentation slides, lesson notes, course books, homework assignments, examinations; this ad-hoc collection may be stored in a wide variety of file formats [11].

The game script collection in the repository includes the games generated. Games can be created, played, and then improved (edited/saved) as needed. The games are organized by broad subject area (professional development, science, arts and humanities, medicine, business, and so on). A taxonomy for serious educational games will be explored to improve this categorization [11].

The current focus and progress of this Summer 2013 term has been the search algorithm and the expansion of the game repository. A visual interface for the game generation component is also in the works for this summer term. See more in Conclusion and Future Works.

**3. Background**

The most common example of AHP is deciding on a car to buy from a list and having different priorities on different aspects of the car such as safety, Cost, seating/storage, warranty. The searcher gives each of these four attributes a “weight” in relation to each other. For instance, safety is twice as important as cost, and cost is a third as important as seating/storage etc, etc.( see Fig 1) Each of the alternative decisions are given a score for their attributes and a weighted score is then calculated using the “weights” of the attributes. For example, if a car scores a 7/10 in safety it would earn 70% of the possible points for that category. The weighted scores are then added up and the contestant is given an overall score. The contestants are then sorted into a list with the highest/ best choice at the top. For slight randomization, we randomly generate a number, if it is divisible by three take the highest weighted component, else continue onto the next component with a new random number. (The best choice is the default case if a number divisible by 3 is never generated).

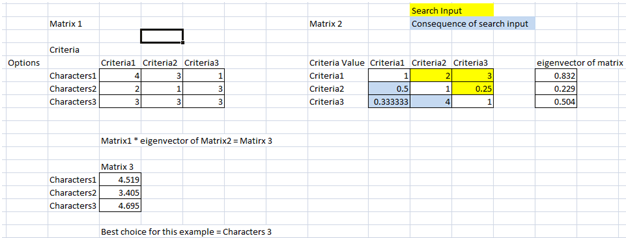


Fig. 2 Original AHP Work

A sample test case from our AHP algorithm has many more criteria and choices to choose from. For instance here is an example from a test case searching for a relevant Subject XML file

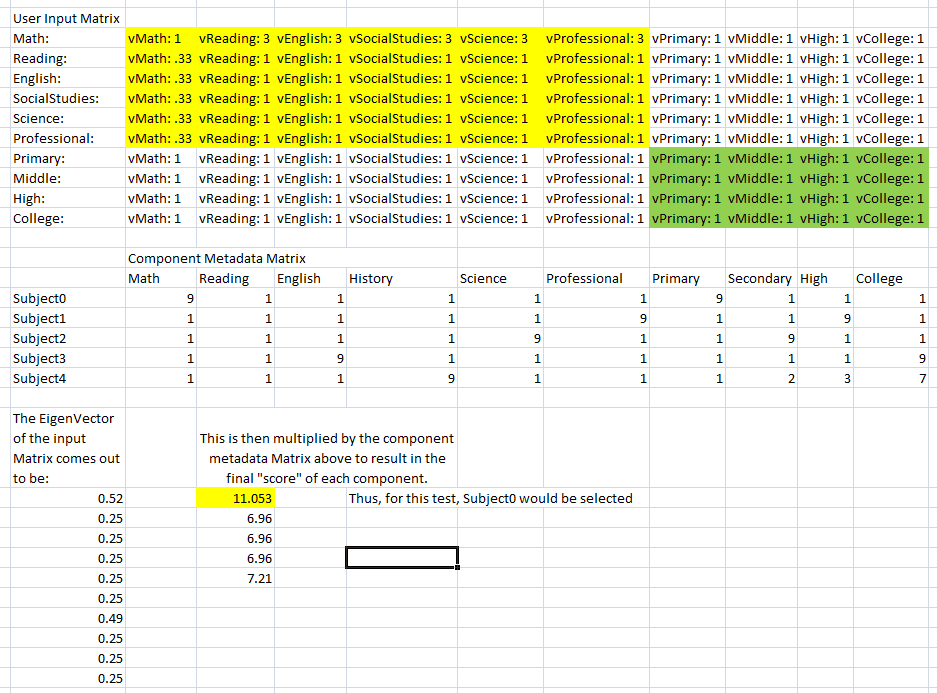


Fig. 3 Additional Work

**4. Game Components**

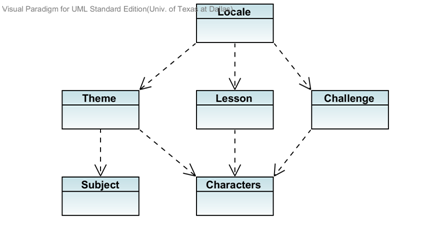


Fig. 4 Component Dependencies Overview

The overall structure of the game is broken down into five different layers or components. The diagram above is a high level view of how the components merge to form the desired game. This diagram will be reference throughout while we discuss each layer individually. The search meta-data is not demonstrated in this diagram but it will be visible in the individual components. The meta-data for each component will be stored in a separated XML file. In order to implement the AHP algorithm each element will be given a physical number that will be fed into the algorithm for calculation.

**4.1 Character Layer:**

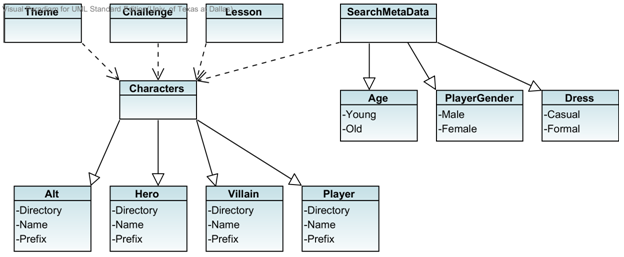


Fig. 5 Character Component Overview

The character layer contains names and assets called from the repository that can be used for each of the four supported characters. The four characters are: Alternative (Alt), Hero, Villain, and Player. The character component is directly used by the theme, lesson, and challenge components. Their dependency can be seen in both the diagram above and the layer overview.

As seen in the diagram, the character component contains 3 different search elements and 6 total sub-elements. Age, PlayerGender and Dress allows user to find the perfect set of characters they prefer. And older crowd can focus on dress and age while the younger generation may want to pick their characters based on PlayerGender. Either way these categories provide a quick and easy way to find what you are looking for.

**4.2 Lesson Layer:**

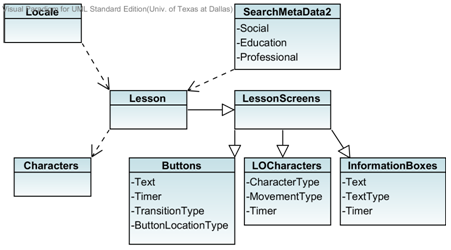


Fig. 6 Lesson Component Overview

The lesson component encompasses a single lesson which directs the student to learn some objective. All objects populating the screen are provided by this component. Such objects include: buttons, textboxes, and characters. The locale will then call this component in order to piece together the lesson screens. (See locale for more detail)

The lesson meta-data is currently very general. We currently do not have enough games to be more specific with our search. For testing purposes we have the 3 broad elements you see in the diagram: Social, Education, and Professional.

**4.3 Challenge Layer:**

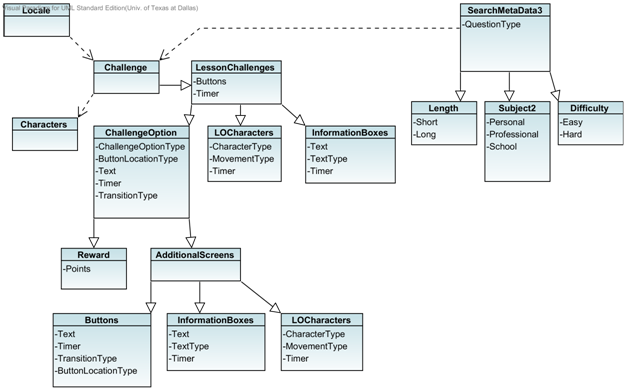


Fig. 7 Challenge Component Overview

Represents the challenge presented at the end of the lesson and is currently limited to a multiple choice quiz of varying length. Like the lesson component, this component populates the challenge screens and failure screens (Additional Screens) with buttons, textboxes, and characters. The locale component then brings it all together to complete the screen. This is also where points and a timer can be handled.

The meta-data for this component contains length, subject, and difficulty. All of which have subcategories which are assigned number values as mentioned before. This component, like the lesson, is still very general because we are in early stages of the xml component repo. As more challenges are created more search elements can be added.

**4.4 Locale Layer:**

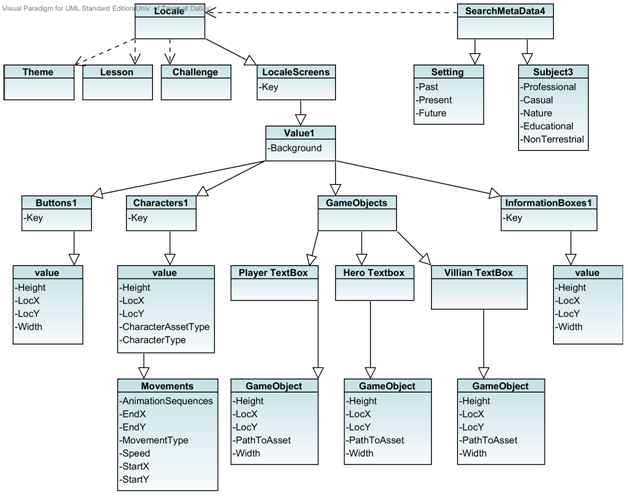


Fig. 8 Locale Component Overview

The locale component is what ties everything together. It provides the background and dialog bubbles for the lesson screens, challenge screens, failure screens, introstory screens, and outrostory screens. The locale is also responsible for the location and movement or animation of all the objects (characters, textboxes, dialog bubbles, buttons) that are created by the individual components for the same screens (lesson, challenge, failure, introstory, outrostory screens).

Locale meta-data is characterized by a specific setting and subject. The current subcategories for setting are: Past, Present, Future. The subcategories for the subject consists of: Professional, Casual, Nature, Education, Non Terrestrial.

**4.5 Subject Layer:**

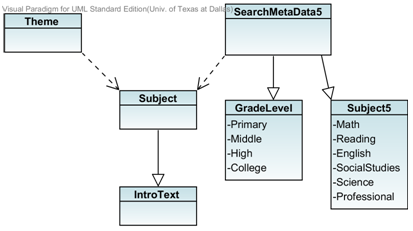


Fig. 9 Subject Component Overview

The subject component describes the overall subject that is being taught and any introductory text. The theme component is supposed to be dependent on this component however it still needs to be implemented.

Meta-data for this component is currently very broad because our component repository is still very small. Someone looking for a game over a specific topic will need to be able to search for more than just the subject or grade level. This component’s meta-data will be updated at a future date.

**4.6 Theme Layer:**

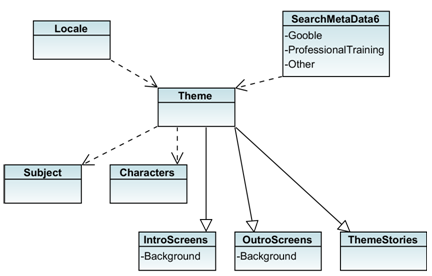


Fig. 10 Theme Component Overview

The theme layer contains the story aspect of the game. This component provides the game objects for the introstory screens and the outrostory screen. These two screens are then put together by the locale. The intro and outro screens are fully built from this theme component. The game objects are created and then placed on the background that is also provided by this component. Due to the size of the Theme component it is presented in three diagrams. The first of which is an overall look at the component and then the IntroScreens, OutroScreens, and ThemeStories are expanded more below.

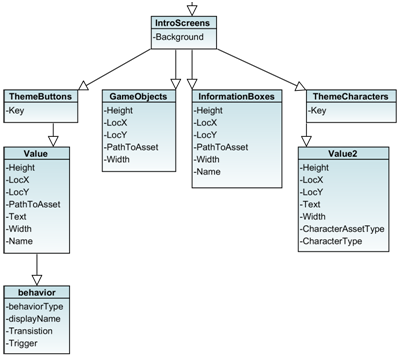


Fig. 11 IntroScreen Overview

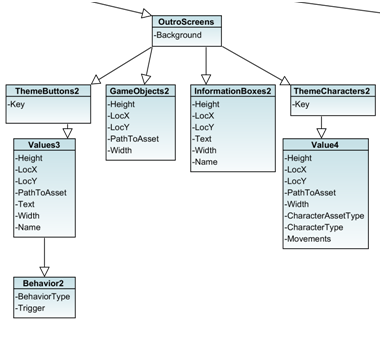


Fig. 12 OutroScreen Overview

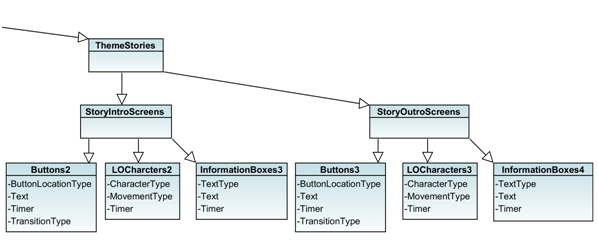


Fig. 13 ThemeStories Overview

The theme meta-data theoretically can be interchangeable with all the components. The component adds creativity and appeal to the game. The current searchable elements are Gooble, Professional Training, and Other.

**4.7 Selection Process**

The selection process is still in the works. It currently takes input from text files that have been edited by the user and then selects the recommended components based on the criteria provided by the text files. There are currently no major dependent variables being used by the selection so our AHP algorithm may return a game that contains a science lesson with a history challenge. Future works include adding these dependent variable that will ensure the fluidity of each game.

**5. Validation**

Test Case #1 is the output of the search algorithm with all current defaults selected (the input matrixes were all 1s),

Test Case #2 was given these goals for its input

Lesson: Educational

Locale: Past, casual setting

Subject: Science, Primary level

Theme: Gooble game if possible

Challenge: Short, easy, school setting/ based

Characters: Young, male

|  |  |
| --- | --- |
| **Original Space Game Story Intro Screen.** |  |
| **Test Case #1 Story Intro Screen** |  |
| **Original Math Game Lesson Screen** |  |
| **Test Case #1 Lesson Screen** |  |

Fig. 14 Test Cast #1

|  |  |
| --- | --- |
| **Original Math Game Story Intro Screen** |  |
| **Test Case #2 Story Intro Screen** |  |
| **Original Space Game Lesson Screen** |  |
| **Test Case #2 Lesson Screen** |  |

Fig. 15 Test Case #2

**6. Conclusions and Future work**

Serious educational games (SEGs) are recognized as valuable educational tools, with significant potential to provide immersive, engaging and fun environments for students across diverse domains. Developing these games, however, remains challenging. A component-based semi-automated game generation approach has the potential to support educators in rapidly creating games. Here, we have introduce our proposed SEGCB approach, which has an interactive wizard user interface, a game generation engine, and a repository of components, game scripts, and game assets. The game components proposed in this preliminary work are the theme, characters, subject, locale, lesson, and challenge. The components have been captured in XML; loaded, composed, and output as a game script [11].

The ability of mix and matching game components are now being tested and have shown great promise. As shown with a few examples in the two test cases the ability to combine different components is successful but not without limitations. One limitation with the components is that each game component should only be paired with components of a similar subject or theme. Our test cases were used without those dependencies which created a game that is not responsible. For example, a space lesson should not paired with the challenge about unrelated vocabulary. As our repository of games continue to grow so will our ability to create a variety of different games pertaining to one subject or theme.

The addition of the search class/component and XML repositories (metadata and component) have been a major focus in this term and it is progressing. Outside of the actual search feature, tools have also have been made to simplify the addition of new components along with their metadata to their respective repositories. This allow the each repository to be updated by running one tool. The search class currently takes input from hard coded text files but it is a future goal of this term is to implement a user interface for the game engine.

**7. References**

[1] Gee, J. P. (2007). What Video Games Have to Teach Us About Learning and Literacy. 2nd Edition, Palgrave Macmillan.

[2] Heineman, G. T., Councill, W.T., Component-Based Software Engineering – Putting the Pieces Together, Addison Wesley, 2001.

[3] Klopfer, E., Osterweil, S., & Salen, K. (2009). Moving Learning Games Forward. Massachusetts Institute of Technology, USA: The Education Arcade.

[4] S. Longstreet and K. Cooper, “A Meta-model for Developing Simulation Games in Higher Education and Professional Development Training”, IEEE 17th International Conference on Computer Games, 2012.

[5] Lopes, R. and Bidarra, R. (2011). Adaptivity challenges in games and simulations: a survey, *IEEE Trans.on Computational Intelligence and AI in Games*, Vol. 3, Issue 2, pp. 85-99.

[6] Sanford, R. (2006). Teaching with Games: COTS games in the classroom. Innovating e-learning: Transforming Learning Experiences online conference.

[7] Scacchi, W. (2011). Modding as a basis for developing game systems. In *Proceeding of the 1st International Workshop on Games and Software Engineering*, pp. 5-8.

[8] Shaffer, D. W. (2007). How Computer Games Help Children Learn. Palgrave Macmillan.

[9] Szyperski, C., *Component Software*, Addison-Wesley, 2 edition, 2002.

[10] Walker, A. and Shelton, B.E. (2008). Problem-based educational games: connections, prescriptions, and assessment. .*J. of Inter. Learning Res*.,vol.19, no.4, pp. 663-684.

[11] Daconceicao, Ricardo, Carsten Locke, Kendra Cooper, and Shaun Longstreet. “Semi-automated Serious Educational Game Generation: a Component-based Game Engineering Approach.” 2013. *Microsoft Word* file.