Design and Assessment of a Pedagogical Approach to Teaching Computer Science to Game and Animation Students

Research Proposal
By: Derek L. Manns
March 17, 2018
Version 11
Abstract:
The proposed research will explore methods for teaching elementary computer science (CS) concepts through approaches in game art and design. Recent changes in game design software include the ability to model systems. Consider Blueprint in the Unreal Engine or the node-flow model plugins for Unity. These game model types create a potential convergence with software engineering modeling, leading to the notion that through game design, one may learn some aspects of CS. But, which CS concepts and how to teach these concepts while covering game design? Can CS principles and game design principles be co-taught within the same class? How can this education be formally assessed? Answering these questions involves the core of our research. An example of a CS principle is iteration. Iteration can be constructed using control structures in a text-based programming languages. It can also be specified in a timeline based modes such as Unreal Engine’s Blueprint or Unity3D’s uScript. The use of matinee in Blueprint or animation view with uScript will provide student with the proper context that may be relevant to their creative interests. This research will focus on two approaches to enhance learning of computer science principles by leveraging video-game design and development. Method 1 will list the computer science concepts to create through the game engine toolset. Method 2 involves the in-game learning of the principles through gameplay experiences. We will use formal assessments to evaluate learning outcomes and compare each approach. To assess research outcomes, we will create in-game scenarios to assess the player’s engagement as it pertains to learning, retaining, and testing their knowledge through development building gameplay.
Introduction

The abstract techniques involved in computer science methodologies can be arduous when educating students. Creating a sound lesson plan would be the logical path to a successful learning experience for any novice. Wu, Chen & Lee point out that “computer science concepts are abstract and difficult to understand for learners, especially for high school students. Learners’ interests in getting deeper understanding of how the computer science concepts work has been decreased even among undergraduate students with a computer science major (Lattua, Meisalob & Tarhioc, 2003).” (Wu, Chen & Lee, 2009). Employing viable examples to support an effective strategy is key to a successful solution. To provide a thorough case study, we will examine game development toolsets within 3D game engines such as Unreal Engine and Unity3D and provide computer science concepts as they pertain to its functionality. A concept such as a while loop can be difficult for novice coders to understand. A while loop executes until the condition is met. The instructions within the loop define the conditions. The construction of the loop and its condition can be assembled through gameplay elements. A crafting system is a great way to learn as you play format. It encourages the player to seek out the components needed to create code snippets in order to level up their skills and progress through the game. Another aspect is the state machine and how it relates to the matinee component in the Unreal Engine. The state machine provides the value of character animation components and their state.

The engaging experience in video games through its gameplay, visual cues, and audio submersion can provide an effective learning environment for students. Video games influence users with various gameplay types from single player action-adventure to multiplayer online battles. The gameplay types mentioned can be an enlightening aid to the learning experience.
Battistella and Von Wangenheim support this with their preliminary results "show learning for game players, compared both through pre- and post-tests differences and improved performance on relevant final exam questions when compared to students who did not play the game."

(Battistella, Wangenheim, 2016).

Computer science fundamentals will be introduced to first and second year non-Computer Science undergraduate students through game development toolsets. Students will be exposed to computer science principles in a 3D video game environment. 3D game engines can be used in a creative and effective way with instructional materials to teach computing principles through integration of game-design methods applied to teach these computing fundamentals. Research by Battistella and von Wangenheim, indicates that there is a “lack of synthesized information to guide development for teaching computing competencies through video game-based instructional methods.” (Battistella, Wangenheim, 2016). Some literature reviews exist but no true methods have been applied. Lessons can be designed to influence the user’s cognitive skills based on their overall engagement within the tools used to create games. Through environmental exploration and social interaction assistance with an NPC\(^1\) and/or other human players, the student can complete lesson objectives. Levels of achievement will be based on a similar role-playing game environments such as encounters with other characters, skill improvement, and skill tests. Character encounters will be designed to help the student seek assistance with their problem. Improving skills will come in the form of repetition or practice. Skill tests will be done in the same manner as a “Boss Fight”\(^2\) or end of objective assessment. Designing these in game interactions are important to engagement and success. If the gameplay, as it pertains to the lesson plan, is not designed well there is a chance of losing the interest of the

---

\(^1\) NPC Non-Player Character otherwise known as an artificial intelligence character

\(^2\) Boss Fight is when a player in an RPG scenario have to defeat a Boss to level up their character's skill
student/player. The strength of in class lectures to provide proficient guidance will be the foundation. Sheldon supports this by saying “If the software fails to entertain, it can be even more boring than the worst lecturer. If the software concentrates too much on fun, it risks obscuring the learning objectives” (Sheldon, 2012). The objectives prepared for lessons will be immersive and captivating to the player. To support our research we will utilize the computing attitude survey (CAS) to help in the learning assessment of the student through focus groups. The assessment will be done in game to keep the student engaged in the entire experience. The tested knowledge and problem solving skills are computer science specific.

Objective:

We will develop learning principles from an in game design perspective enhancing the computer science objectives to solidify the engagement factor. Adding the proper visual element to a lesson can be equally engaging as a well-planned lecture in a classroom setting. Interacting with your student body, providing visual cues, and clear examples are key to this success. Along with well-spoken direction and a commanding and engaging voice, a student can learn just about anything. This is precisely the reason video games would be a great parallel to a well thought lesson plan and accompanying lecture. Video games can set a tone with the proper audio and visual cues that immerse the player in the same manner a theatrical movie with the exception of controlling the outcome. Immersion can be designed to teach through visual layout, encounters with artificial intelligence also known as NPCs, or a specified gameplay style such as exploration through role-playing or flying a ship through our solar system. Practicing the theory of design and applying it to this research effort is supported by Rouse when he says
“A designer/programmer is able to have an idea for some gameplay and then instantly be able to attempt to implement it exactly how she wants it.” (Rouse, 2001). Non-Computer Science students, I feel, will come to understand computational thinking through these gameplay experiences when designed to properly convey the content. Rouse supports this with “A designer/programmer will also often be able to better understand the technology involved in a project, and be able to see what is easily accomplished and what is not.” (Rouse, 2001). The typical path to learning code is to install an IDE and perform the tasks assigned by writing code and printing or displaying the result through a command line format. There is no interaction just the code, execution, and output.

For the non-computer science majors, using video games to learn the fundamentals of programming can be visually advantageous. The advantage allows the user to see results in real time and reinforce or help with the specified content. The video game Wu’s Castle created by

---

3 Integrated Development Environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.
Battistella and Von Wangenheim, instructs the player to instantiate programing functionality allowing the player to execute and visually see the results in real time (Battistella, Wangenheim, 2016).

In figure 2 we see what the perspective paths are possible for an approach to the potential lesson plan. While discussing a proxy for time, a mental model of the processing can be formed. The theoretical model we will present can be derived from programming code, a flow chart, visual scripting as it pertains to the unreal game engine called Blueprint, and real world analog representation. The computational study performed will explore the CS1335 Computer Science I for Non-majors course what can be done to engage the student in computer science concepts through design.

The UT Dallas 2016 Undergraduate Catalog describes the course as follows “Introduction to object-oriented software analysis, design, and development. Classes and objects. Object

The figure example of a simple while loop interaction is much more appealing and engaging which was created in the unreal Engine 4 environment. This is the visual scripting language Blueprint used in Unreal Engine 4. The previous example represents what will be utilized in a 3D environment with well-designed and engaging encounters to fully immerse student players in computer science fundamentals. The gameplay is important in its affirmation to apply the design of the lesson plan. The use of a game engine with specific genre attributes such as Unity3D or Unreal Engine 4 will be considered based on programming language and associated visual design tools. We will answer this question by avoiding the learning curve during the design and development process. Content creation based on computer science concepts, will be vital when
designing a proper interactive lesson plan. “On the undergraduate level, it is expected that students acquire computing competence, mainly, on the cognitive levels of knowledge, comprehension, and application in accordance to Bloom’s taxonomy of learning objectives” (Battistella, Wangenheim, 2016). Once this is established and presented through gameplay


1. Instructional Design Perspective

Curriculum design through a video game’s associated gameplay will be compared to the delivery of similar principles in the form of standard classroom lectures. Listening to the lecture through an in-game narrator as the visual cues are presented fall in line with most digital interactions. A companion and/or on screen text is the default manner in which instructions are presented to the player. When teaching any subject, in this case computer science concepts, engaging non-CS students is key to the success in implementing strong design principles. Basawapatna, Koh, and Repenning state that “Through game creation, learning key points of educational game theory, and play testing games with target audiences, students could better understand what was effective and what was not in terms of educational game creation.” (Basawapatna, Koh, Repenning). Teaching through digital/video games can be used at scale by enhancing accessibility of teaching materials to a broader and potentially more diverse audience. How do
we bridge the gap between Computer Science, Designers, and Artists? Provide standard level of competency for designers and artists to provide an understanding of the implementation of design and art. This is another motivational notch that can be presented by the instructor to provide clarity to why some computational skills need to be acquired. Jason Hayes makes his point in a development article on code and art in the game development industry and he states: “As their interest and experience level grows, some technical artists grow into more lower-level languages such as C# and C++. This gives them greater ability to write compiled plug-ins, such as exporters, and tools outside the content creation packages.” (Hayes, 2008).

2. Assessment Perspective
Being the most important aspect, integrating assessment directly into the gameplay, we can provide feedback in real time to the student. Real time integrated assessment is part of the gameplay and more effective from engaging perspective. Designing with this in mind supports Dr. Val Shute’s approach. The broad justification is that everyone should have a level of computational literacy. That is why the focus is on non-CS majors for an engaging learning experience. It can have broader implications: Such as K-12 and Adult informal learning due to the changes in our evolving world of technology for everyday living and career choices. This will be compared to classroom based teaching methods and learning experiences. Usually chosen by those who are only interested in science and technology, a student’s path to learning is experienced by the standard textbook and an integrated development environment or IDE. The integrated assessment approach students may encourage students to spend more hours “building their computational literacy skills”. Through role-playing game experiences students may strengthen their self-efficacy with respect to computing concepts. To prove the acquisition of knowledge with computer science principles through gameplay will be explored. The
Computing Attitudes Survey Version 4 will be implemented into the gameplay as an objective to ensure 100% participation. The participation will come with a reward in the form of character building experience points or XP\textsuperscript{4}.

\textsuperscript{4} XP also known as Experience Points is the point system used in role-playing games and other character building games to build the character's progression through gameplay.
Computing Attitudes Survey (v4)

Here are a number of statements that may or may not describe your beliefs about learning computer science. You are asked to rate each statement according to the following rating scale:

Strongly Disagree  
Disagree  
Neutral  
Agree  
Strongly Agree

Choose one of the above five choices that best expresses your feeling about the statement. If you don’t understand a statement, leave it blank. If you understand, but have no strong opinion, choose “Neutral”.

Please choose the appropriate response for each item:

<table>
<thead>
<tr>
<th>Item</th>
<th>Strongly Disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. After I study a topic in computer science and feel that I understand it, I have difficulty solving problems on the same topic.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>2. Errors generated by computers are random, and when they happen there’s not much I can do to understand why.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>3. If I want to apply a method used for solving one computer science problem to another problem, the problems must involve very similar situations.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>4. I can usually figure out a way to solve computer science problems.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>5. When I solve a computer science problem, I break it into smaller parts and solve them one at a time.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>6. I do not spend more than five minutes stuck on a computer science problem before giving up or seeking help from someone else.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>7. There are times I solve a computer science problem more than one way to help my understanding.</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

Figure 4 Computing Attitudes Survey V4
3. Methodology and Design Principles

The goal is to deconstruct game design principles to ascertain the efficacy of specific learning objectives. As Lee Sheldon points out in his book The Multiplayer Classroom: Designing coursework as a game, “designing classes as games has produced some very interesting results in classrooms ranging from middle school through university-level courses in a variety of disciplines” (Sheldon, 2012). Once the Instructor is able to teach, assess, and apply the specific subject matter, the student is placed into the equation. Once the game has been designed to incorporate specific computing concepts and content, and this is includes integrated assessment, only then will it be important to examine the impact on the student’s engagement and learning.

Student

1. Engage

Capturing a student’s attention will come from more than the intended lesson but the incorporated gameplay. Affective assessments of a student’s self-efficacy and their interest in obtaining computing knowledge and skill will determine and the value it holds in the minds of the present day student. What will be the social and interpersonal impacts / benefits from game-based learning when compared to traditional methods? Interacting with NPCs and proper narration designed to guide a student will provide a great foundation. Providing High-Score / Badges / Compare Decision Stats - Motivational feedback as it pertains to the lessons and skill advancement will be explored as part of the design process. Sheldon supports this when he says “We’ll explore how to approach grading as attrition, rather than focusing on letter grades, and how to tap the social networking memes and tools already used by students.” (Sheldon, 2012). By classroom standards, students would be a grade after the fact but can now be done in real
time and allow the student to correct and improve their skills without the grading wait time and feedback.

2. Learn
Designing the objectives are crucial to the success of the engaging lesson plans. Navigating the lesson plan through gameplay will be assessed. Once lessons are completed they are assessed based on the following:

a. Was the problem solved?

b. The number of attempts.

c. Time taken to complete.

More assessment options can be added as successful data is retrieved to bring the classroom completely to the digital space. Our baseline for the measure of success: what is the old way and compare that to the gameplay experience? Barnes says that “Lepper and Malone have investigated the most important factors in making educational games fun, listing as most game design books do the importance of challenge, or the balance between easy and difficult, in engaging learners in games, but highlight the need to design in activities that help learners address and revise their misconceptions [7].” (Barnes, Powell, Chaffin, Lipford, 2008). If a student makes a mistake and they revisit the problem once they realized their mistake, they can correct and advance to the next level.

3. Comprehend
Understanding the content provided in the lesson plan through integrated gameplay without extraordinary effort, is the main goal. How do we measure learning of the lesson plan through the student’s experience?

i. Gameplay decisions as it pertains to the lesson objective.

ii. The path taken to solving the problem for a linear or nonlinear design.

iii. Complete objectives and prove a clear understanding of the material.
If a student can engage, learn, and comprehend the material provided by the instructor, the principles are proven to be successful. But to ensure these perspectives achieve success, the student must have computational thinking skills needed to learn the material and play games in a 3D environment. Guzdial says “Computer science involves a new way of thinking about problem-solving: it’s called computational thinking, and it’s about understanding the difference between human and artificial intelligence, as well as about thinking recursively, being alert to the need for prevention detection and protection against risks, using abstraction and decomposition when tackling large tasks, and deploying heuristic reasoning, iteration and search to discover solutions to complex problems.” (Guzdial, 2016). The computational thinking will need clear assessment integration into the gameplay for a successful outcome.

**Methods:**

Role-playing games and its associated character building scenarios can be used to create an engaging lesson in computer science or is it better suited for another academic subject? There is also the turn-based approach versus real-time RPG scenarios. Turn-based gameplay is based on each player rolling dice to determine their next decision. Real-Time Strategy games (RTS) can be considered due to its real time decision gameplay. Multiplayer scenarios can be used to test your knowledge with another real-time user or teacher over a network or peer-to-peer. While First-Person Shooters can fall in line with the multiplayer scenarios, it has a single player element that will allow the user to experience a small classroom one-on-one immersive learning experience. While RPGs and RTSs are not strictly tied to 3D environments, we will assume this moving forward. These genres have the most flexibility in their ability to meet content creation standards. A list of genres are listed in the appendix.
We will design games to educate the user through standard practices while keeping her engaged. We can apply familiar gameplay scenarios previously mentioned or create unique education based designs to assist educators with the video game design process. We need to narrow down the learning styles of users in order to create a game learning environment for all user types. This can be explored when Tobias and Fletcher state “how different players use space and how we can create dialogs between users and a space’s unique qualities” [28]. The user can play through a scenario to experience and understand the lesson’s objective. Understanding the method and repeating this by way of the lesson’s gameplay is one way of deductive proof. There are a number of existing video games that teach players particular principles that can be translated to real world scenarios. James Gee details 5 principles as they relate to different gameplay types. Active, Critical Learning Principle, Design Principle, Semiotic Principle, Semiotic Domains Principle, and metalevel thinking about Semiotic Domains Principle. They all refer to a level learning and involvement during gameplay engagement [7].

Two major ways to present lessons to the user is in a linear and nonlinear fashion. Create a linear level or world to guide the user in a sequential manner. Progressive objectives are not met until the prerequisite is met. Co supports this when he says “Players have no choice in how to play the game...Linear levels typically flow in a straight line from the start to the exit.”[5]. Create a nonlinear level or world which allows the user to complete objectives in, what is referred to in the video game world as, an open-world environment. In an open world environment the user can roam around without being restricted like a train on railroad tracks. Co also states “The branches could go in different directions, or they could meet up later on.”[5]. A linear track would be the logical choice for introductory to computer science principles. Structure and specific memory management are great examples to support this method. Role-playing games allow the player to
improve themselves through a leveling up approach. This approach can be tied into the linear
guide to completing objectives. Real-time strategy games allow players to strategize their
progress against real players or non-player characters otherwise known as artificial intelligence.
While RPGs can be played in third or first person views, it is typically played from the third
person point of view. Therefore allowing the first-person genre the overall view of the digital
space. Perry says “The character is the player’s extension into the game world.” [19]. The
specified genres have their advantages and disadvantages but can all be used to convey the
lesson plan set forth by the instructor.

The nonlinear approach can be used as an open-world environment for users to explore, learn,
and reinforce their computer science knowledge. Open-world environments work best with
nonlinear designs and can be found in RPG, FPS, and Simulations (SIMS) scenarios.
Following a similar method executed by Lee Sheldon, classes will be designed in a game world
with character design (building) traits to engage and motivate the student to accomplish the
specified objectives designed for our research efforts. Sheldon supports this with the following
“Designing classes as games has produced some very interesting results in classrooms ranging
from middle school through university-level courses in a variety of disciplines.” [22]. The design
approach provided will inherit RPG elements increasing the engagement with character building
methods.

The player will select the tools or objects found in the world necessary to solve a puzzle which is
code based. The abilities acquired through gameplay will be in the form of functions which allow
the player to use them according to the problem encountered in the game. The statistics or Stats
saved in the game are directly related to the assessment of the player’s progress. The Progression
is assessed to provide a conclusive evidence of the player’s success based on what they have
learned through gameplay interactive scenarios. Comparative skills acquired through an experience/level system are successful campaign objectives with health (knowledge acquired), MP Mana Points aka Magic (program functions), and Stamina (can be based on positive assessment). The common character RPG classes, according to the Dungeons and Dragons Player’s Handbook:

Table 1 Classes and Academic subjects are not directly compared

<table>
<thead>
<tr>
<th>Classes</th>
<th>Description</th>
<th>Academic Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barbarian</td>
<td>A fierce warrior of primitive background who can enter a battle rage</td>
<td>Algebra I</td>
<td></td>
</tr>
<tr>
<td>Bard</td>
<td>An inspiring magician whose power echoes the music of creation</td>
<td>Biology I</td>
<td></td>
</tr>
<tr>
<td>Cleric</td>
<td>A priestly champion who wields divine magic in service of a higher power</td>
<td>Calculus I</td>
<td></td>
</tr>
<tr>
<td>Druid</td>
<td>A priest of the Old Faith, wielding the power of nature—moonlight and plant growth, fire and lightning—and adopting animal forms</td>
<td>Computer Science I</td>
<td>Introduction to object-oriented software analysis, design, and development.</td>
</tr>
<tr>
<td>Fighter</td>
<td>A master of martial combat, skilled with a variety of weapons and armor</td>
<td>Digital Art</td>
<td></td>
</tr>
<tr>
<td>Monk</td>
<td>A master of martial arts, harnessing the power of the body in pursuit of physical and spiritual perfection</td>
<td>Fine Arts</td>
<td></td>
</tr>
<tr>
<td>Paladin</td>
<td>A holy warrior bound to a sacred oath</td>
<td>Game Design</td>
<td></td>
</tr>
<tr>
<td>Ranger</td>
<td>A warrior who uses martial prowess and nature magic to combat threats on the edges of civilization</td>
<td>Geography</td>
<td></td>
</tr>
<tr>
<td>Rogue</td>
<td>A scoundrel who uses stealth and trickery to overcome obstacles and enemies</td>
<td>Graphic Design</td>
<td></td>
</tr>
<tr>
<td>Sorcerer</td>
<td>A spellcaster who draws on inherent magic from a gift or bloodline</td>
<td>History</td>
<td></td>
</tr>
<tr>
<td>Warlock</td>
<td>A wielder of magic that is derived from bargain with an extraplanar entity</td>
<td>Physics I</td>
<td></td>
</tr>
<tr>
<td>Wizard</td>
<td>A scholarly magic-user capable of manipulating the structures of reality</td>
<td>Trigonometry</td>
<td></td>
</tr>
</tbody>
</table>
Design approach:

Visual programming design in game approach

Recently, there have been many 2D based visual scripting game engines such as GAMEBLOX [15] that teach programming. I submit that the utilization of RPG elements and associated sound cues in a 3D environment is more engaging. 3D worlds provide more immersive feedback to the student/player. Virtual and augmented realities allow the same learning environments as 3D worlds but in intense FOV⁵ environments! The 2D based engagement is not completely lost due to the use of a well-designed User interfaces. They provide some of the engaging elements involved in its gameplay. But using the Unreal Engine 4 3D game engine with their visual scripting language, allows the developer to create gameplay mechanics without C++ programming. This is an important factor for non-computer science majors and their connection to code needed to solve design problems. The visual nature of the system allows real time monitoring of the applied functions. Teaching students the visual system in the game would be redundant so why design in game components that represent the programming components needed to instantiate a function. Using a linear approach to this visual design scheme will allow the instructor to guide the student with the in game components needed to complete an objective. Simple effective lessons design to engage students will support this approach.

The toolset included in the Unreal Engine 4 game engine will be utilized to create a crafting system⁶.

---

⁵ FOV Field of View in first person video games or field of vision is the extent of the observable game world that is seen on the display type.
⁶ Crafting System
**Text based command line in game approach**

This approach can seem redundant but can cut out the middleman so to speak, an IDE (Integrated Development Environment) in this case, because the lessons are introductory in nature. The system can be built to simulate the compiling of simple code in game. Engaging the student user in a way that teaches and entertains without the feeling of being in a classroom is key to the research. The visual scripting and text based approach can have their place in a 3D learning environment with the right gameplay design attached. Visual scripting can be designed so that it is integrated within the gameplay environment, and can be used for in-game activities like puzzle-solving. The text based can be as well for beginner style or relevant gameplay style as well such as computer terminals or smartphones as an example. This would support the stealth assessment point made by Val Shute as it pertains to teaching a student without them realizing they have actually learned or provided feedback about their interactions. Shute states “*Stealth assessment is seamlessly woven directly into the fabric of the instructional environment to support learning of important content and key competencies. This represents a quiet, yet powerful process by which learner performance data are continuously gathered during the course of playing/learning and inferences are made about the level of relevant competencies (see Shute, Ventura, Bauer, & Zapata-Rivera, 2009)*”[29]. The assessment the gameplay will embody is important to the fluidity of the design, its gameplay and lessons they manage.
Conclusion:

The study of computer science fundamentals in a video game through a 3D environment would be a new approach to this subject matter. The design methods integrated into the gameplay will be key to the success of the students based on ability to learn, retain, and reiterate the given material. The student will construct a game with elements of the game in the same 3D environment. Alternatively, the student will play a role-playing character in-game. By doing so, the user can learn as many Computer Science principles applied to a given game.

We propose a design approach in an RPG environment to allow instructors to integrate lesson plans and grade students based on their level and completion status. The level or leveling up as it is referred to in RPGs, will be compared to a grading system. The game engine toolset will be used to teach specific computer science fundamentals followed by objectives to complete to prove learning competency. It will be at the discretion of the instructor and their implementation of the subject matter. The lesson can be in a linear and nonlinear forms. Artificial Intelligence integration for feedback and or lecture will be explored within the lesson. Sikora and Hitan say “A design will typically be considered good if it fulfills the requirements in a meaningful way.”[27]. Their choice in design styles provided can executed to a level the instructor feels is needed to engage their student body’s computational skills.
References:


