Blender and Unreal Engine Character Design and Behavior Programming for 3D Games

Sreekanth Dekkati

Assistant Vice President (System Administrator), MUFG Bank, New York, **USA**

Corresponding Contact:
Email: sreekanthd041987@gmail.com

Manuscript Received: 10 Nov 2020 - Revised: 18 Dec 2020 - Accepted: 25 Dec 2020

ABSTRACT

A software game is a program used for entertainment and severe purposes that can be applied to many fields such as education, business, and health care. These more severe uses can apply to a variety of domains. The software game development technique is distinct from traditional software development due to the multidisciplinary nature of the game development methods, which include elements such as sound, art, control systems, artificial intelligence (AI), and human factors. The fundamental software engineering principles allow game creation to achieve maintainability, flexibility, reduced time and expense, and improved design. This study's objectives are to (1) evaluate the current level of research on the process of game development software engineering and (2) draw attention to aspects of this process that require additional investigation by researchers. In the research, we utilized a methodology that consisted of a comprehensive literature evaluation based on widely recognized digital libraries. The production phase of the game development software engineering process life cycle has been the focus of most research published on the topic. The pre-production phase has followed this. In comparison, the amount of research focused on the post-production phase is far lower than that of the pre-production and production stages. According to this research, developing video games through software engineering has many facets that require more attention from researchers; this is especially true regarding the post-production phase.

Keywords: Digital Games, Model, Game Development, Unreal Engine, Blender

INTRODUCTION

Technology has taken over our everyday lives to the point that we are surrounded by it no matter where we go. We can find gadgets embedded with software in most of them. These devices can execute various jobs quickly and be created to cater to our wants and requirements so that we can get the most benefit out of them (Aleem et al., 2016).

Suppose one focuses on the industry of digital games. In that case, they will discover that it has benefited from this, and according to the Entertainment Software Association (ESA), an organization founded by numerous game creators, the industry made a profit of twenty-three and a half billion dollars (Thaduri et al., 2016). In addition, according to the ESA, on average, 1.7 gamers live in each home in the United States. However, even though the business appears to be doing well, there needs to be more development rules and models that hinder the production of games. Because of this challenge, a different strategy was required.

Software can be defined as any collection of instructions that enables us to interact with technology. Since digital games are software, this definition includes software. In this instance, the software emphasizing enjoyment is a digital game (Thaduri, 2017). Because of this, one could believe that utilizing a software development model is sufficient to design a game successfully; moreover, digital games are not a result of pure engineering; they cannot be developed using a systematic and stringent engineering process. However, digital games are not considered works of pure art either; instead, they result from interweaving multiple disciplines, including art, music, programming, acting, and the management or integration of all these elements (Lal et al., 2018). Because of this, creating a video game calls for a very particular and precise development procedure. Despite this, software development models continue to serve as a valuable resource of principles, concepts, and processes, yet they still need to be improved regarding the particulars of game development.

In addition, the production and development of video games as a means of enhancing the quality of life for people with disabilities is not a novel concept; however, it is a "noble" concept that has, up until this point, counted on some efforts that followed an ad hoc approach rather than a systematic approach that was guided by a well-recognized and well-documented process for creation and production (Lal & Ballamudi, 2017). The Social Tech Booster, found online at http://stb.uninova.pt, is one example of an initiative producing distinct "serious" games and systems towards this purpose. The initiative brings together educational institutions and students in their final year of study in engineering. Through completing their master's thesis, the students are allowed to make a positive difference in the lives of several other people. At the same time, the institutions receive more resources suited to their requirements, ultimately altering the lives of those in the greatest need (Best, 2013).

The issue mentioned before is that there needs to be a development model that can successfully design and develop a game for this purpose. In addition, the process by which the STB initiative is carried out creates further barriers to progress because of the constraints imposed by both parties involved (Thaduri, 2018). A problem that needs to be solved is the knowledge gap between the two parties, which is caused by factors such as the lack of time that the students have to develop the game (they have only nine months to do it), the lack of resources that the institutions can provide, and so on. Using the STB initiative as a starting point, this article proposes a game development model capable of effectively delivering a high-quality game (Ballamudi, 2016).

PROGRAMMING CHARACTER BEHAVIOR

Molding a character's behavior in a game is complicated and requires various tools. Instead of depending on Unreal's physics engine, the game's primary characters use root

motion technology (Lal, 2015). This method enables the animations to control the movement of the character's root bone. Animators now have greater control and precision over the character's movement and interactions with the environment. Using keyframe animation in Blender to produce the gameplay animations and then merging everything in the AnimGraph, a technology that enables us to create complicated animation logic for our characters using nodes and graphs, is how the animations are created. Because we can bundle many states into a single alias and then use it as a condition for transitions or blends in Unreal Engine 5, it is straightforward to keep everything organized. This is made possible by the State Aliases feature, one of the many features that make it possible. We can simplify our AnimGraph and keep from repeating the same logic for each of the different states if we do it this way (Chen et al., 2012).

We also use Masks, additive animations, and cached poses to add further depth on top of everything else. Masks are a type of node that enables us to apply different animations or effects to some skeleton regions while isolating those parts from the rest (Ballamudi & Desamsetti, 2017). If we want the upper body to move in another animation from the lower body, we can use a mask. Additive animations are layered on top of another animation, so altering the pose is achieved at the end. Using an additive animation, for instance, we could add tilting to the flying or the running and varied positions while the character is aiming when they are shooting (Thaduri, 2019).

There are nodes in the AnimGraph known as cached poses. These nodes allow us to save a posture at a specific location in the graph and reuse it later. For instance, we can use a cached pose as a transitional tool to transition between two different positions seamlessly. Using root motion in a multiplayer game is not a good idea since it can cause synchronization issues and latency problems. Using root motion in a single-player game is not a problem. However, when used in a game with only one player, it can offer quite a few more layers of realism and allow animators to "direct" and control the movement of a figure (Desamsetti & Mandapuram, 2017). Now, non-playable characters are different. Blueprints are used instead of behavior trees in creating the game's primary adversaries, for example. Blueprints are incredibly versatile and powerful tools that may be used to design anything from straightforward triggers to intricate artificial intelligence systems (Ma, 2013).

However, what about the crowd NPCs, the people we see doing business in the various cities? They take a hint from Unreal's Matrix Demo and the plug-in Mass AI system, which is a tool that enables us to create large-scale crowds of autonomous agents that can navigate complex settings. They take a cue from Unreal's Matrix Demo and the plug-in Mass AI system. Because of this structure, the non-playable characters (NPCs) navigate the city by following these predetermined paths while deftly avoiding impediments, such as the main character or other NPCs (Lal, 2016). We are letting them make their own decisions.

We combine Mass AI with our own NPC randomization blueprint, which is a mechanism that enables us to generate a wide variety of NPCs with their personalities. Booleans and data tables are used to pick various values for each NPC, such as gender, race, haircut, dress, accessories, etc. This allows the system to function correctly. After that, the values are used in the creation script to alter the NPC's appearance and qualities. For instance, we can decide the gender of the NPC by using a boolean, then use a data table to choose a random name, hairdo, and outfit for that gender, and finally update the textures and

parameters in the materials. Because of this, rather than having to design each NPC manually, we may generate hundreds or even thousands of unique NPCs.

This approach is a massive assist in populating the game's cities because it enables us to construct realistic and diverse crowds of NPCs, giving the environment a more vibrant and immersive feel overall. We can also utilize this method to design NPCs that fit the theme and atmosphere of each city, such as futuristic, cyberpunk, steampunk, or any of the other themes above. An illustration of this system can be seen in this location. Remember that the system still needs to be done and that the final product will have additional age possibilities, outfit selections, and materials.

We also decided to combine behavior trees and root motion to simulate the movement of animals and other unique NPCs that are free to explore the open world. A behavior tree is a graphical representation of the decision-making process that an artificial intelligence agent goes through. They are made out of nodes that can represent various things, including actions, conditions, sequences, selectors, and decorators. By mixing behavior trees with root motion, we can provide animals and NPCs with realistic movements and dynamic behaviors. This allows us to develop more complex characters. Ultimately, every character is unique, and to make them behave precisely how we want, we will need to use the appropriate tools from the toolbox provided by Unreal.

COMBINING BLENDER AND UNREAL ENGINE

Blender is often where we do all our modeling for environments, except for minor items. This means the level design is created entirely within the Blender program. Since we have already coded the character's movement, it is much easier for us to make it directly in a modeling application, which is a technique that might only be encouraged if we are intimately familiar with the character's movement. However, doing so is much simpler (Turner et al., 2016).

As we previously stated, the initial layout and the building were created by us in Blender. After that, everything is imported into Unreal in distinct parts, and after that, we begin the process of scene dressing with smaller items such as lighting, benches, trees, and other elements along the same lines. When it comes to the appearance of characters and other props in Unreal, we find it helpful to have a visual reference of what they will look like. To see textures and other features, we use our shaders in Blender. This gives us much creative freedom. The shaders in Blender and Unreal are essentially the same thing from a technical point of view.

Importing animations is the component that poses the least amount of difficulty. If an animation does not function as intended, reimporting it into Unreal is as simple as conducting a drag-and-drop operation on our computer. This is typically a procedure that we carry out several times. In addition, Unreal has a set of tools for modifying animations, such as changing their pace or adjusting the curves in facial mocap. Therefore, Unreal is always open to making a few tweaks here and there (Dekkati & Thaduri, 2017). However, since Unreal has streamlined the import procedure, the shortest option is reimporting it from Blender. However, it is essential to remember that the programs use different units of measurement and even axis orientations, so it is necessary to be aware of these differences. As a result, we will need to configure the scene in Blender and the export and import settings so that everything runs smoothly. Once that step has been completed, switching from Unreal to Blender is a relatively simple process.

We also collect motion capture data for our work using Rokoko Studio (with the Smartsuit and Smartgloves) and the Live Link Live application, specifically designed for facial mocap. In addition to it, we make use of other applications, such as Substance 3D Painter. Because of this, we are accustomed to switching between software regularly and developing a smooth and efficient routine.

SETTING UP THE SCANNING EFFECT

The scanning effect is a function that, when activated, will allow us to view more information about the planet and the characters in the game, providing details and hints that were previously hidden. However, putting it into practice is pretty simple, with the most challenging component being the visual aspect of it (Chen et al., 2019). Configuring a flip-flop node in the blueprints is necessary to produce the scanning effect. A flip-flop node is called if it switches between its two different outputs every time it receives an input. In this scenario, the input consists of the button turning on the scanning mode. The first output causes the activation of a post-processing shader. This shader applies an effect to the final displayed image by utilizing a timeline, a node that allows us to animate a value using curves. This is followed by triggering a boolean variable named Scan, which can only take on one of two values: true or false (Yoon & Kim, 2015).

The scanning effect relies heavily on the post-processing shader to do its work. It produces a mask that spreads outward from the camera's location to "the end" of the world, covering everything that comes into contact with it. The mask can alter the look of the items behind it, including adding a grid, an impulse, a chromatic aberration, or even outlines on the buildings. Controlling the visibility of specific objects in the world, such as additional user interface, character information, mission cues, and so on, is another use of the mask (Dekkati et al., 2019). These components are only revealed whenever the mask is positioned to cover them; otherwise, they remain concealed in the background. This results in a scanning effect that is both dynamic and participatory.

Scan is the name of the boolean that is used to keep track of whether or not the scanning mode is currently active. It controls the visibility of specific items worldwide, such as additional user interfaces, character information, mission hints, etc. When set to true, it enables the post-processing shader and allows the extra elements. They are rendered inoperable once that property is set to a false value. Additionally, the boolean is used to connect with other blueprints or scripts that may require information regarding the status of the scanning mode, such as whether it is active or not. If we push the input button again, the flip-flop node's second output will deactivate the post-processing effect through another timeline. It will also turn off the Scan boolean, returning everything to its initial, standard state.

We are going to talk about the "city generation" here. It is similar in concept but more complicated since it also involves the integration of level streaming, which is a technique that enables us to load and unload parts of a significant level dynamically, depending on the location and direction of the player in the level. It is similar but significantly more complicated.

OPTIMIZATION

Our absolute favorite aspect is optimization, and we employ various tools to ensure everything goes well. On a laptop outfitted with a GTX 1060 GPU, 8GB of RAM, and an i7

processor, the game plays flawlessly with all settings set to Ultra at 1080p, getting 30-50 frames per second (FPS). When we consider this, it is clear that optimization is one of the most important things for us to focus on (Aleem et al., 2016).

To begin, the game does not use Lumen, VSM, or Nanite, even though these are fantastic tools that make the work of developers much more accessible. When these tools are used, however, it is possible that the game will not function as intended on systems with a middle-range to lower-end hardware configuration or even on consoles from an earlier generation. As a result, we first turned off these options (although the appropriateness of this move dramatically depends on the nature of our project and our goals).

Instancing and Level of Detail (LOD) are used extensively throughout the game's cities and world. Instancing is a technology that allows numerous copies of the same mesh to be rendered with only one draw call. This brings the CPU overhead down, which ultimately results in improved performance. LODs, which stands for Levels of Detail, are simplified representations of a mesh presented at various distances from the camera. These mesh versions are rendered in different ways depending on the distance. This lowers the stress on the GPU, which in turn increases performance. On the other hand, collision detection is maintained as straightforward as practically practicable because complicated collisions can result in substantial problems (Liang & Feng, 2012).

The master material for buildings is another method that can be utilized. The buildings in the game have a great deal of intricacy and typically contain between 12 and 16 different materials within each structure. This could result in issues with the draw call. A draw call is a command that tells the GPU to render a geometry batch with a specified set of state changes (Dekkati et al., 2016). A draw call is also requested to keep this explanation as straightforward as possible (Ballamudi, 2019a). As a result, making excessive draw calls might slow the rendering process and impact performance. To address this issue, a new material was developed that combines all of the existing textures and materials into a single material, essentially including a Material Atlas function into the material itself (the work of PrismaticaDev served as inspiration for the development of this material). A material atlas is a texture comprising numerous sub-textures, each of which may be accessed using various UV coordinates (Ballamudi, 2019b). This is a more technical explanation of what a material atlas is. If our scenes have many different materials, we should use this strategy. Our GPU will be grateful.

The rotation of items is the subject of yet another entertaining trick. Certain cities feature hundreds of things that rotate at the same time. The traditional method for accomplishing this goal uses animations and skeletal meshes, even though they can affect performance; alternatively, it uses rotating components or tick events, even though both of these concepts struggle when dealing with hundreds of spinning objects. To solve this problem, we devised a shader that gives the impression that the object is rotating even though it does not move in that manner. A visual technique analogous to the movement of plants in response to the character or the wind.

CHALLENGES IN GAME DEVELOPMENT

Only 16% of projects are finished on time and under budget, as indicated by the findings of a recent data gathering conducted by Pretillo. In addition, it is reasonable to deduce from the data collected that the difficulties that occurred more frequently (over fifty

percent) resulted from ineffective project management and inadequate requirement collection (Desamsetti, 2016a).

The production of video games is fraught with difficulties, and the failure rate is high; nonetheless, many of these issues have been addressed and resolved by the computer software industry. To find solutions to problems, it is necessary first to recognize and comprehend the challenges at hand, as is the case with most situations. The most significant competitors are as follows:

- Diverse Resources Video games are the product of the collaboration of a large number of specialists in a variety of fields. As the project progresses, managing all of these is an increasing challenge.
- Diverse Resources Video games are the product of the collaboration of a large number of specialists in a variety of fields. As the project progresses, managing all of these is an increasing challenge.
- Scope: Because there was not enough time spent developing a feasible and viable design and planning, the size of the project is continually growing as new features are introduced. Adding features one after another with little thought will eventually result in the installation of unrealistic features.
- Publishing: Bringing the game to the industry may be difficult owing to the lack of investment or outdated technologies. This is because the video game industry is highly competitive and fast-paced.
- Management: Managing many resources while ensuring the project stays on schedule necessitates effective communication between all participants and vigilant supervision.
- Third parties and emerging technologies: The intense rivalry in the gaming business drives the ongoing creation of innovative and cutting-edge new technology. Managing this challenge could be problematic easier if the appropriate technology is utilized.
- The organization of a team is a complex process since ensuring that all team members are in check, thinking the same way, and working for the same common goal is challenging.
- The project's success may be directly correlated to the process selected for its development. Having a solid grasp of the procedure is also essential.
- These issues can also occur with the STB initiative and must be considered when designing this new paradigm.

Even though Christopher M. Kanode and Hisham M. Haddad have devised several solutions, such solutions need to match the criteria for the STB initiative. However, knowing the presence of these difficulties as well as the significance of them is sufficient to enable one to make conclusions and get the model ready to deal with them.

DIGITAL GAMES' DEVELOPMENT MODEL

The problems with the STB described earlier are addressed, and a potential solution is presented in this article in the form of a new model for game development (Desamsetti,

2016b). In other words, the creation of high-quality video games that are effective via the use of a methodical procedure rather than an ad hoc strategy to enhance the lives of people who have disabilities through the collaboration of organizations that care for children and undergraduate students (Pirovano et al., 2016).

The following game development models, each consisting of five stages and developed with the information gained from software development models and the usual issues that arise during the development of games, while also bearing in mind the entities involved in the STB project, are offered.

Planning Stage: The purpose of this stage is to prepare for game production. First, develop the game concept. Three key activities are suggested:

- Brainstorm: Develop game concepts by compiling feature ideas and conducting market research to identify comparable or related games.
- The second activity, Meetup, involves meeting the "client ."In this situation, daycare. The goal is for students and institutions to share ideas, hopes, and features. Proper documentation of this meeting is crucial for the game development process.
- Set-up, the final planning process, refines the game based on prior outcomes and prepares for development.

All game features must be documented and a development timeline established, starting with critical features. Maintain these documents in a portfolio for transparent project supervision.

Design Stage: The Design stage includes creating and revising the game design and the initial prototype. Like the preceding stage, it has three key activities: Appointment, the initial activity, bridges the knowledge gap between institutions and students, preventing communication breakdowns, and allowing students to obtain crucial information. Throughout go-along visits, students observe personnel of the institution throughout their daily routine, receive information on available resources and patient limits, and gain insight into their target audience (Cardoso et al., 2017). In the second activity, the prototype is developed, containing all game instances and adjustments to the earlier planning to account for new features. The final phase, re-evaluation, involves creating Game Design Documents with the final design, including features, planning, and the initial prototype (Desamsetti & Lal, 2019). Remember to keep all completed documents from all stages in a portfolio.

Development Stage: The Development stage is the main procedure for creating the game. This stage uses an iterative approach for game creation. With each iteration, a new feature is added, tested, and reviewed. A feature is produced and introduced to an existing project before being tested. At this stage, the development team conducts functional tests, not fun ones (Desamsetti, 2018).

An inspection of the completed project follows testing. Initially, a professor conducts the inspection, followed by the institution. The initial inspection assesses if the institution needs to clarify specifics and affirm or deny development decisions, ensuring the game remains on schedule (Deming et al., 2018).

Evaluation: The evaluation step evaluates the game's functionality and enjoyment appeal. This stage involves simple testing assessing both functional and enjoyable features (Ballamudi, 2019c). The testing process involves three parts, each with assessment sheets.

The project has three phases: the first involves the development team and those who are entirely familiar with it, the second involves those who are less familiar, and the third involves those who fit the desired target, childcare institution members, and other professionals.

Deployment: At this point, the "deployment to the masses" and the institution are under this stage's purview. The development team must choose a way of deployment that the institution may easily access while also considering the resources at their disposal. The rollout should occur in two stages: initially, it should target the institution and then move on to the broader public.

CONCLUSION

Two games were developed using the proposed paradigm for validation. To validate the suggested model, the success of these games was compared to earlier games. In the planning stage, the development team created a basic game with additional features. To make the game more engaging, an activity was included. However, during the appointment activity during the design stage, the childcare facility stated that the game should be simple without extras. Extra features outside gaming. The goal was to Replace and offer a fresh method of therapy. In this scenario, game simplicity was essential. So that children with problems already Do not lose primary teaching in actions other than those that boost their standing, degrading all work. The childcare institution and professor accepted a simple prototype. The development stage featured four iterations. The first iteration, core one difficulty level, was created for games. And few words. The subsequent iterations increased the difficulty and words, all according to Childcare institution specifications, and evaluated and met with their therapists. First, the professor considered and then First by STB students, then members of the childcare center, and then patients. The findings showed a simple, shallow game. However, it fits the establishment. The deployment only targeted institutions since it was a focused game, but some changes happened.

REFERENCES

Aleem, S., Fernando, C. L., Ahmed, F. (2016). A Digital Game Maturity Model (DGMM), *Entertainment Computing*, *17*, 55-73. http://dx.doi.org/10.1016/j.entcom.2016.08.004

Aleem, S., Capretz, L. F., Ahmed, F. (2016). Game Development Software Engineering Process Life Cycle: A Systematic Review. *Journal of Software Engineering Research and Development*, *4*(1), 1-30. https://doi.org/10.1186/s40411-016-0032-7

Ballamudi, V. K. R. (2016). Utilization of Machine Learning in a Responsible Manner in the Healthcare Sector. *Malaysian Journal of Medical and Biological Research*, *3*(2), 117-122. https://mjmbr.my/index.php/mjmbr/article/view/677

Ballamudi, V. K. R. (2019a). Artificial Intelligence: Implication on Management. *Global Disclosure of Economics and Business*, *8*(2), 105-118. https://doi.org/10.18034/gdeb.v8i2.540

Ballamudi, V. K. R. (2019b). Road Accident Analysis and Prediction using Machine Learning Algorithmic Approaches. *Asian Journal of Humanity, Art and Literature*, *6*(2), 185-192. https://doi.org/10.18034/ajhal.v6i2.529

Ballamudi, V. K. R. (2019c). Hybrid Automata: An Algorithmic Approach Behavioral Hybrid Systems. *Asia Pacific Journal of Energy and Environment*, *6*(2), 83-90. https://doi.org/10.18034/apjee.v6i2.541

Ballamudi, V. K. R., & Desamsetti, H. (2017). Security and Privacy in Cloud Computing: Challenges and Opportunities. *American Journal of Trade and Policy*, *4*(3), 129–136. https://doi.org/10.18034/ajtp.v4i3.667

Best, B. J. (2013). Inducing Models of Behavior From Expert Task Performance in Virtual Environments. *Computational and Mathematical Organization Theory*, *19*(3), 370-401. https://doi.org/10.1007/s10588-012-9136-8

Cardoso, T., Sousa, J., Barata, J. (2017). Digital Games' Development Model. *EAI Endorsed Transactions on Serious Games*, *4*(12). https://doi.org/10.4108/eai.8-12-2017.153399

Chen, G., Li, B., Tian, F., Ji, P., Li, W. (2012). Design and Implementation of a 3D Ocean Virtual Reality and Visualization Engine. *Journal of Ocean University of China. JOUC*, *11*(4), 481-487. https://doi.org/10.1007/s11802-012-2112-6

Chen, S., Thaduri, U. R., & Ballamudi, V. K. R. (2019). Front-End Development in React: An Overview. *Engineering International*, *7*(2), 117–126. https://doi.org/10.18034/ei.v7i2.662

Dekkati, S., & Thaduri, U. R. (2017). Innovative Method for the Prediction of Software Defects Based on Class Imbalance Datasets. *Technology & Management Review*, *2*, 1–5. https://upright.pub/index.php/tmr/article/view/78

Dekkati, S., Lal, K., & Desamsetti, H. (2019). React Native for Android: Cross-Platform Mobile Application Development. *Global Disclosure of Economics and Business*, *8*(2), 153-164. https://doi.org/10.18034/gdeb.v8i2.696

Dekkati, S., Thaduri, U. R., & Lal, K. (2016). Business Value of Digitization: Curse or Blessing?. *Global Disclosure of Economics and Business*, *5*(2), 133-138. https://doi.org/10.18034/gdeb.v5i2.702

Deming, C., Dekkati, S., & Desamsetti, H. (2018). Exploratory Data Analysis and Visualization for Business Analytics. *Asian Journal of Applied Science and Engineering*, *7*(1), 93–100. https://doi.org/10.18034/ajase.v7i1.53

Desamsetti, H. (2016a). A Fused Homomorphic Encryption Technique to Increase Secure Data Storage in Cloud Based Systems. *The International Journal of Science & Technoledge*, *4*(10), 151-155.

Desamsetti, H. (2016b). Issues with the Cloud Computing Technology. *International Research Journal of Engineering and Technology (IRJET)*, *3*(5), 321-323.

Desamsetti, H. (2018). Internet of Things (IoT) Technology for Use as Part of the Development of Smart Home Systems. *International Journal of Reciprocal Symmetry and Theoretical Physics*, *5*, 14–21. https://upright.pub/index.php/ijrstp/article/view/89

Desamsetti, H., & Lal, K. (2019). Being a Realistic Master: Creating Props and Environments Design for AAA Games. *Asian Journal of Humanity, Art and Literature*, *6*(2), 193-202. https://doi.org/10.18034/ajhal.v6i2.701

Desamsetti, H., & Mandapuram, M. (2017). A Review of Meta-Model Designed for the Model-Based Testing Technique. *Engineering International*, *5*(2), 107–110. https://doi.org/10.18034/ei.v5i2.661

Lal, K. (2015). How Does Cloud Infrastructure Work?. *Asia Pacific Journal of Energy and Environment*, *2*(2), 61-64. https://doi.org/10.18034/apjee.v2i2.697

Lal, K. (2016). Impact of Multi-Cloud Infrastructure on Business Organizations to Use Cloud Platforms to Fulfill Their Cloud Needs. *American Journal of Trade and Policy*, *3*(3), 121–126. https://doi.org/10.18034/ajtp.v3i3.663

Lal, K., & Ballamudi, V. K. R. (2017). Unlock Data's Full Potential with Segment: A Cloud Data Integration Approach. *Technology & Management Review*, *2*(1), 6–12. https://upright.pub/index.php/tmr/article/view/80

Lal, K., Ballamudi, V. K. R., & Thaduri, U. R. (2018). Exploiting the Potential of Artificial Intelligence in Decision Support Systems. *ABC Journal of Advanced Research*, *7*(2), 131-138. https://doi.org/10.18034/abcjar.v7i2.695

Liang, L., Feng, G. (2012). A Game-Theoretic Framework for Interference Coordination in OFDMA Relay Networks. *IEEE Transactions on Vehicular Technology*, *61*(1), 321-332. https://doi.org/10.1109/TVT.2011.2176356

Ma, S. G. (2013). The Research of Next-Gen Game Engine Virtual Reality in Actual Project. *Applied Mechanics and Materials*, *443*, 39. https://doi.org/10.4028/www.scientific.net/AMM.443.39

Pirovano, M., Mainetti, R., Baud-Bovy, G., Lanzi, P. L., Borghese, N. A. (2016). Intelligent Game Engine for Rehabilitation (IGER). *IEEE Transactions on Computational Intelligence and AI in Games*, *8*(1), 43-55, https://doi.org/10.1109/TCIAIG.2014.2368392

Thaduri, U. R. (2017). Business Security Threat Overview Using IT and Business Intelligence. *Global Disclosure of Economics and Business*, *6*(2), 123-132. https://doi.org/10.18034/gdeb.v6i2.703

Thaduri, U. R. (2018). Business Insights of Artificial Intelligence and the Future of Humans. *American Journal of Trade and Policy*, *5*(3), 143–150. https://doi.org/10.18034/ajtp.v5i3.669

Thaduri, U. R. (2019). Android & iOS Health Apps for Track Physical Activity and Healthcare. *Malaysian Journal of Medical and Biological Research*, *6*(2), 151-156. https://mjmbr.my/index.php/mjmbr/article/view/678

Thaduri, U. R., Ballamudi, V. K. R., Dekkati, S., & Mandapuram, M. (2016). Making the Cloud Adoption Decisions: Gaining Advantages from Taking an Integrated Approach. *International Journal of Reciprocal Symmetry and Theoretical Physics*, *3*, 11–16. https://upright.pub/index.php/ijrstp/article/view/77

Turner, W. A., Thomas, B., Casey, L. M. (2016). Developing Games for Mental Health: A Primer. *Professional Psychology: Research and Practice*, *47*(3), 242-249. https://doi.org/10.1037/pro0000082

Yoon, D-M., Kim, K-J. (2015). Challenges and Opportunities in Game Artificial Intelligence Education Using Angry Birds. *IEEE Access*, *3*, 793-804. https://doi.org/10.1109/ACCESS.2015.2442680

--0--