GRAPHICAL RANDOM MONSTER GENERATION

Vandana Sharma

A thesis submitted for the degree of Master of Science in Computer Games

Supervisor: Dr. Richard Bartle

School of Computer Science and Electronic Engineering

University of Essex

August 2019

ABSTRACT

Several highly successful games have shown the increasing demand for procedurally generated content in the game industries. Use of procedural content generation is not new; it is a method of using the algorithm to generate content which is typically generated manually. Many approaches are used to generate contents by a procedural method like search-based method, using pre-made parts, noise-based method of content generation. Procedural contents required to make the game more engaging and interesting for gamers or users. To fulfil that demand more procedural content generated tools are required which can generate reasonable contents. Generation of procedural content saves money by decreasing the demand and affords of game artists, which is profitable for game industries.

This research is about developing a tool using procedural content generation method to create random monsters. This is done by writing the algorithm to generate random numbers and select the image according to the selected random number. Research is based on the previous research done by the researchers on procedural content generation method. This tool is implemented using existed monster images as input and generates new random monster images in output which can be saved for further use.

CONTENTS

Table of Figures

I. INTRODUCTION

Presence of Computer game is increasing in our lives. Every day millions of people are playing computer games. According to the survey [1]. The US Entertainment Software Association (ESA) claims that more than 68% of American homes play computer or video games in 2009. Game for gamers contents many things like 2D or 3D objects, puzzles, Characters, gameplay, maps, which play a significant role in their entertainment. But over the past century, the population of gamer is growing exponentially, and this becomes a problem for manual labour to match the quality and the amount of game material according to the gamer's community requirement. Procedural content generation is used to address this challenge by the automatic generating game content. Like other assets, Monsters are important factors keeping players engaged and attract in the playing world. The demand for new monsters keeps increasing while manual production of monsters is expensive. Three centuries of studies, many techniques exist today to generate many kinds of game content procedurally. Pseudo-random number generation is one of the easiest and earliest solutions to the generation of procedural game material [1].

The research work is done to develop a tool which generates new monster images automatically using procedural generation and taking existing monster images as input. Purpose of the tool is to generate the monster image to give an idea of monster appearance which satisfies the constraints imposed by the artist and return the gamer with interesting monsters. The output from the tool can be used by character artists of games to visualize the appearance of monsters which can be further modified by the artist to use in the game according to their requirement.

From years researchers are interested in procedural content generation technique of content generation which is discussed in the background section of this document [1]. A different method of procedural generation is discussed in the background section. Some of the work done by researchers is mentioned like using procedural content generation, Artificial Intelligence and other methods.

The main objective of this project is explained also the methodology used to develop this tool is described in detail within this document. System requirements for the development of the tool like a programming language, other software, and the design of the tool as well as the detailed process of development of the tool is also described.

Development of tool faced many problems and after solving the entire problems occurs during development, even after passing unit tests performed on this tool, it generates random monster and give effective results. Feedback collected from users manifests that developed tool is useful for game artists. Many improvements can be done to make the user experience of this tool better.

II. BACKGROUND

Procedural Content Generation (PCG) is the use of a formal algorithm to produce game content typically generated by a person. The interest in PCG in the gaming sector is not new, from many years the method of creating random content have been in interest [1]. The first procedural game was not a computer game and it quite old. In the early stage, motivation for Procedural content generation was to provide repayable experience and helping players in their creativity [2] For example, the game "Quantum", it is a strategy board game in which the original piece setup is randomized and any original puzzle piece setup is valid [3].

Many different approaches used to generate different contents for video games [2]. There are many examples to create realistic-looking trees (SpeedTree) to games using whole universes (Elite Dangerous, No Man's Sky). The goal is not only to find entirely new generating methodologies but also to try to find better ways to control the random processes involved and to make the result more reliable and more variable at the same time [1]. Togelius et al. distinguish between content created by the user and the generation of procedural content. Which states that a person may create procedurally generated content if that person follows a specified procedure [4].

About procedural content generation, early thinking was that the computer could provide an unbiased interpretation of the mechanics of a game. Procedural content generation using computers are unbiased described by The Trolls and Trollkin supplement of pre-generated monsters. They describe its contents as being “created by the computer program to eliminate bias and error” [5]. Similarly, monsters for Dungeons & Dragons are generated by a computer program. But the computer is just a machine with a lack of imagination and the ulterior motive that the human might have. The computer can only generate contents blindly.

a) PRE-MADE PARTS

There are many techniques used for Procedural Content Generation. One of the simplest and widely used technique is to use pre-made parts and combine them in the way to generate something new. The same technique was used by Hao Wang in 1961 [6] for content generation, Hao Wang invented tiles called Wang Tiles which is used to generate levels in the game. These tiles are usually a four-colour square on edges. A set of such tiles are arranged as it is to form a pattern without any rotation or reflection. The adjacent tiles must be of the same colour. This method used by Hao Wang to create a different level pattern in for the game. There is a similar method called Occupancy-Regulated Extension used in Super Mario.

b) SEARCH-BASED

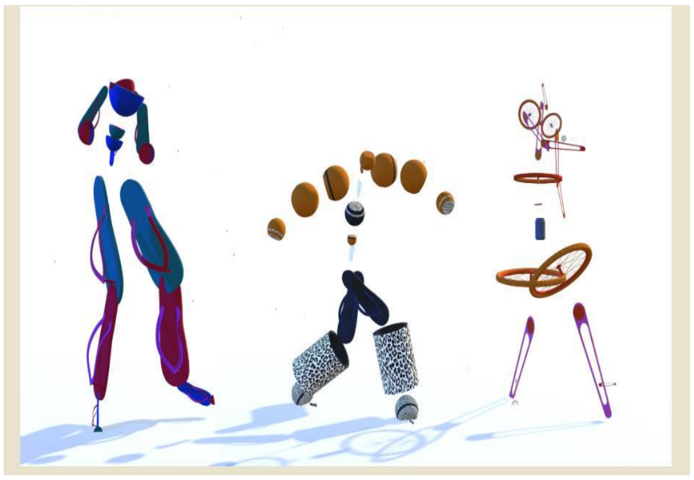
Another method of procedural content generation is searched based method. This method is used to search the optimal solution to a problem throughout the space. But space can contain any kind of solution and finding a meaning full solution needs proper encoding. We can take an example of this method. In a tree structure, Valtchanovet al. [7] coded their methods where a node represents a partial level and an edge represents a connection. Other searched based approach is shown by Steve Dahlskog, Julian Togelius [8] where they demonstrate that how to combine practical game design patterns and procedural content generation to produce game levels that resemble a particular style of design and present the outcomes of a series of studies using a platform game standard. They also presented searched based approach for the game that reflects levels as micro pattern sequences and meso patterns searched. Abstracting common design pattern from the same level is meso pattern were using a bit of original human design pattern from an existing game is called micro pattern. In the level encoding of Super- Mario the same method was used.

c) GRAMMER-BASED

To create a wide verity of different content like vegetation, buildings, levels, missions in the game a method can be used called a Grammar-based method of procedural content generation. This method is consisting of some set of rules which are applied until the termination condition reached to get the result. The method of step-by-step content generation is known as a constructive method. One of the categories of the constrictive algorithm is Cellular automata, which was discovered in the 1940s, but in 1970 a breakthrough bought by John Horton Conway. Cellular automation is a structure of self-organization that consists of a periodic cell grid where each cell has several states, initial state, some set of rules and next state. Appling function gives different pattern as a result which depends on the initial state, set of rules and the number of iterations [9]. This method is used to generate caves by Johnson et al. [10] and to generate landscape and dungeons. The implementation of this method is simple and it can be used to generate infinite levels in real-time.

d) NOISE -BASED

Every natural object consisting of some kind of noise in it, so, by generating noise in the existing data set, new data can be generated. Noise is called a series of Random Number and noise generator as a function which creates random data set. To give a natural look to an object, noise can be added into it. But noise can’t be added randomly to the object there should be some algorithm to make the addition of noise sensual. For example, if a character needs to design by using random objects then noise can be added to select object randomly but it’s not possible to place objects anywhere in the frame, to make it look like a character is should be placed at some particular places.



In figure 1, Random objects are using to create a humanoid character but random objects are placed using algorithm at a particular place not all over the frame which makes the collection of the object look like a humanoid character..

In 1983, Ken Perlin used an algorithm to generate first noise generate using procedural generator [11]. In games, noise generation can be used to make a difference using heights in the landscape. A game called Minecraft used this technique of noise generation

Figure 1: Character design using random objects

III. RELATED WORK

There is an insatiable thirst for more and new game material in the ever-growing market for computer games. As the demand is increasing for games, the fact that more money is needed for modern game development is exacerbated by the reality. In addition to marketing expenditures, developers and game designers are assigned one of the largest components of the game budget. Researchers are innovating new methods every day to produce procedural content to reduce the cost and efforts of designers. There is some work done by researches in the same field to find a method which can make games more interesting and engaging which can profit game artists and industries.

a) PROCEDURAL CONTENT GENERATION

Game content with restricted or indirect user input is created algorithmically [12] is known as a procedural content generation. The main requirement for the content produced by the procedural method is that it should be meaningful when used in the games. PCG relates to computer software capable of creating game content on its own, or with one or more human players or developers. In business games, the generation of procedural content has gained growing attention. Diablo [12] is an action-playing videogame containing the creation of the maps, number and placement of objects and monsters.

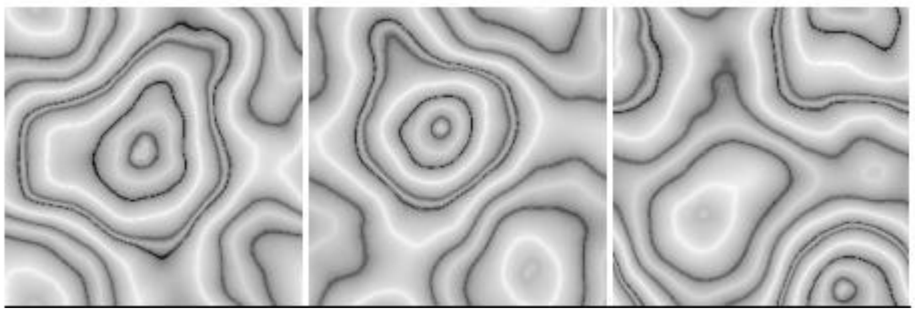


Figure 2: Procedurally generated image [13]

Xenija Neufeld, Sanaz Mostaghim and Diego Perez-Liebana [13] discussed automatic generated game level generator which is designed to read arbitrary 2D games description written in the Video Game Description Language(VGDL). Many different approaches are used to generate levels for video games and the most common is the search-based approach. These approaches have some limitations like these approaches show good result in the level generation for the particular game they are designed for. With same search space, faintness function and content representation they cannot generate new content effectively if the game mechanics, rules or aim changes. They used Answer Set Programming (ASP) and Evolutionary Algorithm to generate new content like maps, dungeon and puzzle games of GVG- AI framework and latter optimiser for level difficulties. This is used to specify the required properties of the content and search space without restricted to any particular game rules. In this method, the logical expression is used to define game mechanics and content structure and constraints and reduce search space and results in no use of faintness function. This method aims to generate better-designed levels with maximum difference which can be appreciated by skilled players.

Creating a new combinational game is easy as compare to predict the game which can interest human players. Every year number of designers produces the number of games but its not possible to test and commercial enquires about all and predict their success. Cameron Browne and Frederic Maire [14] developed a tool which can automatically measure the quality of the game and reduced the development time which benefits both designers and industry by quickly detecting flaws and reducing extensive play-testing needs. This was done by describing a framework called Ludi, which is a system for playing, measure and synthesis the combinational games within the scope of its GDL. Ludi GDL is game descriptive language structure with hierarchical and well-defined nature. It follows the basics means – play-ends models of the game and include the players and game relationship. This tool is useful for designers as its automated new rule combinational search, suggesting interesting avenues or even produces the complete new meaningful game.

Techniques for generating artificial terrain are a significant facet of graphic applications that try to depict a true or imaginary world. During the last few years, numbers of terrain generation techniques have been proposed with some key constrains. Modelling techniques depends on skilled designers and their time and efforts to obtain good results, and cannot be automatically generated [15]

There are some traditional terrain generation techniques: 1) Measuring technique which gathers data evaluated form the real world. It is built using satellite images and land survey. This is used to minimize the human efforts and produce highly realistic terrain. But this technique is time taking as designs need to search for the real-world data according to the game's requirement. 2) Modelling technique is the most flexible way of terrain generation. The human artist is required to model and sculpt the terrain manually using 3D software. Using this method building of terrains may vary according to the features provided by the editor chosen by the artist, but the principle of building terrain is always the same. This technique gives unlimited control over the design and feature of the terrain. Details of the terrain depend upon the designers and this technique increases the requirement of designers in terms of their efforts and time. 3) In Procedural techniques, terrains are generated programmatically. This is furthered divided into two categories. Physically-based techniques simulate the real phenomena of terrains using effects like erosion of wind, water, thermal etc. This technique is very demanding and can generate realistic terrains, but high knowledge of physical laws is required to use this. Another approach is spectral synthesis. This method is based on the observation of a well-defined power spectrum of fractional Brownian movement (fBm) noise [15].

For generating terrains, Evolutionary algorithm is also used. These are bio-inspired algorithms, where living organisms are compensated by their ongoing survival and the spread to the offspring of their genes. Other is Genetic programming, it is an evolutionary computing (EC) method that solves issues automatically without requiring the user to understand or indicate the solution's form or structure in advance [15]. More specifically, it is a systematic method to solve problems automatically from high- level statement. It never guarantees result as it is heuristic nature.

Miguel Frade, F. Fernandez de Vega and Carlos Cotta [15], developed a technique called Genetic Terrain programming technique which enables designers or terrain programmers to develop terrains according to their artistic thoughts or required characteristics. Their main focus was to speed up big terrain visualization to attain interactive frame rates. They experimented on the desired characteristic of video games' designers, that is, terrain morphological feature perseverance on the ground across various resolutions. It will allow them to adapt the terrain to the processing energy needed without any recurrence of extra algorithms, also assist enhance efficiency in the evolutionary stage [15].

b) CHARATER GENERATION USING ARTIFICIAL INTELLIGENCE

PCG study is primarily based on the game artificial intelligence, where it is described as an artificial intelligence challenge for replicating or replacing professional human creative designs, how to have an AI explanation for the quality of the material produced by it and construct an AI system that can communicate with another.

Artificial intelligence is used to make realistic human-like bots to play a game. To improve realism, researchers are investigating a naturalistic navigation system, human intuitive and limitations, and systems simulating a skilled human player's decision process. Any online gaming survey reveals that human prefer human opponent over computer-controlled non-playing character while playing games, no matter how an advance computer may be. This is because the emotional involvement in the online multi-player games is a crucial part of the fun for the players which make players to enjoy a game on a whole new level. The human player understands the computer just doesn't care. A feeling of satisfaction can hardly be derived from a soulless entity that only performs on a technical level and has no emotions on its achievement or failure. NPC, or bot in an action game, doesn't feel any humiliation and no joy in winning. Robert Zubek and Aaron Khoo [16]claim that social aspect of gaming is ignored, in multi-players games emotions of an opponent are far more important than their skill and intelligence, and suggested to make a bot participate in social interaction along with making a challenging bot which will increase the enjoyability into the game. For this, they experimented on the game Counter-Strike selected a bot Tembot from the game which had some chatting capability. They deleted the existing chat for the experiment and implemented their version and able to generate a bot which can deceive a human player for a decent duration of time.

Similar to the agent in the game Quake and Descent [17], where agent playing controlled by using SOAR, Matthew Roberts [18]created a realistic simulation environment intend of increasing game agent capability. Where they used planning architecture which separates planning and execution similar to James Firby [19]Games agents are only suitable for an act in the particular situation according to given instruction or scripts but not as a general-purpose intelligent agent in general environment as they don’t have general planning and problem-solving knowledge. For this simulation environment was developed called CreatureSpace which was a combination of a realistic environment and intelligent agent. The agent could exist in a realistic environment and can behave according to desire. The agent was tested on fire evacuation simulation. Testing Agents on realistic environment allows the agent to face real-life challenges and allow researchers to a developing agent in sterilise environment.

The success of games like The Sims and Black & White demonstrates that Non-Player Characters ' (NPCs) personalities, moods and interactions is in demand to be brought in focus of gameplay. Similar to Matthew Roberts [17] Brian Mac Namee&Pádraig Cunningham also created an Artificial Intelligent non-player character to interact socially. An intelligent agent architecture component intended to create computer games NPCs. μ-SIC uses several psychological quantitative models to simulate the personalities, moods and relationships of characters. And, Artificial Neural Networks (ANNs) have been used to train and take a specific set of values for the above-mentioned psychological models and to determine what interaction a character should have at a given time.

c) SOME OTHER PROCEDURAL CONTENT GENERATION MAETHODS

Biodiversity has extended since the beginning of life as evolution conquered the ocean, soil and air, inventing innumerable adaptations along with the manner [20]. Current human engineering is one of the examples of evolution which demonstrate human-level intelligence. This creativity is also a common feature in digital evolution and not limited to the biological medium. Daniel Dennett [20] said evolution will occur with the presence of three conditions of replication, variation (mutation), and differential fitness. Simply copy a data structure to is replication, using random element in a data structure is variation and selection can be done by the digital analogy of artificial and natural selection in biological evolution [20]

Many digital evolution scientists can give examples of how their changing algorithms evolve. Similar to natural organisms, sometimes unrecognized bugs in their code produced unexpected changes or involved in unusual behaviours and results with asymmetric those observed in nature and surprise the researchers. Other times, Evolution merely surprises and delights by generating smart alternatives not considered or believed impossible by researchers. In one paper [21], researchers discussed the unexpected results of digital evolution, where some of them are classified into different categories. While applying digital evolution to address practical issues fitness function is chosen to reflect the required search goal. In hopes to get further fitness improvements, ultimately leading to the desired result, Breeding is biased towards people with a high fitness rating. Experiments often overestimate the accuracy of their quantitative measure reflects their fundamental qualitative achievement [21]. The most common way in which digital evolution surprises its professionals is to encounter the differences between what we intended to select and what we have chosen, which gives an unexpected result which is sometimes more useful than expected results. Another example of the creative liberty of human preconceptions of digital evolution about what form a solution should take is that research often learns how to detect bugs in simulations [21]. Another kind of surprise is when evolution generates valid alternatives beyond the expectations of experimenters, rather than upholding experimental purpose. Naturally, digital evolution shares significant abstract principles of choice, variation and heritage, even though there is no assurance that digital evolution will show behaviours and results comparable to those observed in nature.

Adventure games, games which include both puzzle-solving and exploration. One experiment was done by Gabriella A. B. Barros, Antonios Liapis, and Julian Togelius [20], to use open data to create easy adventure games. Various game items produced in this fashion are connected through hints pointing to each other, while extra fake clues and dead ends are added to boost the ultimate adventure game exploration value.

IV. OBJECTIVE

Generating Procedural Content is typically a characteristic of digital games. Procedural content generation is where the agent carrying out this operation is presumed simply to be a computer. Procedural content generation is not just a random generation on content, it is a method to overcome the limits of human creativity. We can also define it as an algorithmic generation of game content which limits the human contribution as well as gives unique content every time as output. Content can be anything like in games, different vegetation, buildings, characters, weapons, story, music, textures, rules, levels and maps. In scholarly studies on artificial and computational intelligence in games, automatic content generation is presently one of the most active subjects. A wide range of techniques has been suggested to produce an even wider range of game content, subject to different goals and limitations [22]. In some game-like Rouge and Elite, it is used to generate dungeons and hundreds of stars respectively and generation of trees and grasses and more for hundreds of different games. The main object of Procedural content generation method is to reduce the demand and efforts of game designers and to generate new content with minimal or no human contribution. Procedurally generated contents can be used easily to develop the interest of the player in any game, without putting more efforts in thinking about designing same contents in various ways which can keep player engaged for a long time in the game. Different types of contents can be generated using a different algorithm to help designers.

“Graphical Random Monster Generator” is also a tool to generate procedural contents using random images of monsters from the existing world. This tool is based on an algorithm to randomly select already separated parts of existing monsters and combine them at a particular position which makes a new monster. “Graphical Random Monster Generator” is developed by using the same technology used to generate procedural content by Hao Wang in making of Wang Tiles, which is the use of pre-made parts to generate new contents. In this tool, new contents are generated combining the pre-made parts like different body-parts of different monsters.

Game developers are always interested in using computer-generated contents as these contents can be kept compressed until it is necessary. After three centuries of studies, many techniques exist today to generate different kinds of game content procedurally. Pseudo-random number generation is one of the easiest and earliest solutions to the generation of procedural game material [1]. As there is a demand for different random generated contents in the game, different random content-generating tool is developed. Which are sometimes similar to some other tool which already exists but with some variations like using different techniques to develop the same content which makes that tool different from existing tools. By using the same technology to develop the same content can also give different output by making a small change in algorithm while developing the tool. Here is an example of similar tool which use to generate different contents for game, “Fantasy name generator”, “Spell and monster card creator”, “Map creator”, “RPG villain generator” [13], where “Fantasy name generator” is a description generator, which generates an almost random description of a humanoid creature, which depending on the decision it can be a quite normal type or a more monstrous form. For example:

“Three wide eyes look at their surroundings from their thin sockets. A small nose rests below, but it's the massive mouth below that takes all the attention. A gummy smile reveals several large broken teeth and a huge tongue. Tiny jagged ears sit on each side of its large, bony head, which itself is covered in rows of small horns and has two bony spikes protruding from the top. Its average chunky body is hunched over. Two thin almost branch-like arms hang at its sides and end in broad hands with curvy fingers, of which it has 10 in total. Its legs are skinny and stand straight, each ending in lean feet. Its body is covered in wood-like skin and its shoulders are about the same width as its pelvis, from which a bladed tail sways back and forth.”

Similar to other content in games, monsters are also in demand while make most of the games, And it is time-consuming to for designers to generate new monster idea every time, so the main idea behind procedural generation is to minimise the manual generated or human-generated content, instead, generating contents by executing well define procedure using computers[1].

This can be used by game artists to get an idea for designing a new monster. Game artist are responsible for generation of different contents in games like, levels, characters, maps, trees, weapons, terrains, etc. similarly, some of the games required monster in video game-like “King of the monsters”, “Godzilla”, “Primal Rage”, “Shadow of the colossus”, etc. Use of monsters in some gameplay an important role to keep the audience or their gamer engaged and attracted towards the game by giving interesting effects to the video game. As in the ever-growing computer games industry, there is an insatiable demand for more and new game content, these tools are generated. Demand for monsters is also increasing in the game industries. Some games require more than one monster in a particular game. To contribute to fulfilling the demand of the monster by reducing time and efforts, this tool is generated. Many game designers and artists are required in the gaming industries rather than programmers. A game development company that could substitute algorithms for some of the artists and designers would have a competitive advantage, as industries could be able to maintain quality while producing faster and cheaper games. While maintaining quality. “Monster Manual” [23]is a book consist of monster information. This tool is developed using graphical images of monsters present in “Monster Manual” There are other tools to generate random monsters but they are a text-based monster, which generates a description of monster randomly by selecting different attributes of the monster.

A theoretical assessment of how world-building extends beyond storytelling, public involvement and the conceptualization and experience of worlds is explained by the Mark J. P.Wolf [24] in this book “Building Imaginary Worlds”.

This research is done to answer the research question that, is Procedural content generation method can generate plausible monster images as an outcome.

V. METHODOLOGY

The work done in the development of “Graphical random monster generator” tool is divided into four different phases. Those four phases are defined as, a literature study phase, a data collection phase, a data manipulation phase and finally implementation phase.

In the first, literature study phase aim was to search about the Procedural Content Generation and its benefits, and the methods of Procedural Content Generation which can be used to generate contents in the game. This phase was to learn about the different techniques used to generate procedural contents and how they are different from each other, where and how these procedurally generated contents were used, the requirements for these procedurally generated contents and the comparison with the manually generated contents for the games. Literature study helped to figure out why this tool (Graphical Random Monster Generator) is useful for game designers and can be used by game designers in game development.

Next, Data collection phase, where different data were got collected according to the requirement for tool development. The research about the different types of monsters was done in this phase which concludes that this tool will be developed to give humanoid monsters only. Most of the data collected during this phase were discarded while modifying data as they were not suitable for the tool. Collection of data was based on the output required from the tool, for example, this tool is developed to generate only humanoid monsters so all the images as data, collected are of humanoid monsters from different games. Images of monster had approx. the same resolution was preferred so that they can merge and give a clear new monster image.

In the phase of data manipulation, every collected monster image was divided into three parts as Top, Body and Bottom. Different parts of images after getting divided from original were modified to get fitted with other parts of from different images, the measurement was done before modifying any images. A modification was not done in excessive amount, so at the same time, different parts of images were closer to their originality as well. Images were divided and modified using Adobe Photoshop software using the basic functionality of this software.

At last, implementation phase, which is consist of both implementation of the algorithm testing of the algorithm using data in it, where the algorithms were developed and compared to each other in order to determine which techniques were best used to address the project's objective and made changes in algorithm according to the results obtained for the testing. For developing an algorithm for working of tool and for the design of the tool, Java programming language was used.

For the development of “Graphical random monster generator”, Waterfall approach was implemented. As for this tool, the main content was existing monster images and they need to be collected and modified to make tool. It is not possible to modify images if found a bug later as it will make a requirement of re-edit all images. So, the waterfall model was used so that the first required task can be completed before going to the next one which was essential for the development of this tool.

VI. SYSTEM DESIGN

1) SPECIFIC REQUIREMENTS

➢ PROGRAMMING LANGUAGE

Tool is designed using Java programming language along with Java AWT (Abstract window Toolkit). Java programming language is used in the development of “Graphical Random Monster generator” tool as Java is designed to have very few implementation dependencies. Irrespective of the underlying architecture, Java applications are typically mapped to bytecodes that can operate on any Java virtual machine (JVM), [25]. Java applications can operate without the need for recompilation on all systems that support Java.

➢ TARGETED PLATFORM

This tool is designed for all platform computer, platforms like windows, mac, linux etc. and developed using JDK version 8. To use this tool installation of JDK 8 or above is required, as JDK is required for running all java application. It's a software package that you can use to create applications based on Java programming language [25]. “Graphical Random Monster Generator” tool is also developed in java programming language.

➢ USER INPUT

In Graphical Random Monster Generator tool, user input is required to make tool interactive with users. So that user can give command to generate random monster images or to save random monster images, User can also save image by different names.

➢ OUTPUT

In the tool, as User Input take input from user and generate new monster image and displays the generated new image of monster on the screen within the tool. Generated new monster image from the tool can be saved by user.

➢ DATABASE

As this tool uses the existing monster images to generate random image output, image of different part of monsters are saved into database inside the tools system file. In this tool output images are also saved into database and similar to input database, output files also saves into database.

2) DESIGN

➢ TARGETED AUDIENCE

As described above, this is a tool designed for all 2D and 3D game design artists. This tool is helpful for the games design artist as it give them prospective of new randomly generated monster from existing monsters, which can be used in new games to make game more attractive and Appling for the gamers.

➢ TOOL DESCRIPTION

"Graphical Random Monster Generator" is a tool developed to use procedural method of generating contents and generate random content for the games. For this tool contents means monster images, as this tool is specifically developed to generate new a random monster Images.

➢ GUI

"Graphical User Interface" (GUI) is one of the important part of any program or tool designed for users. User Interface is used to create user-software interaction as easy as possible.This tool is designed to use by the game design artist so, game artists are users for this tool and to make this tool interact with its user. This tool is consisting of GIUs like Buttons, text box and frame.

Buttons: there are two buttons in this system. First button is "Generate" button which is defined to run program on click and generate random graphical images. Other button is "Save" button, which is defined to save the images generated by the tool using generate button.

Text box: To save multiple images user need to give different name to the generated output image, to take name for monster as input text box is defined in the tool where user can write name for the particular monster by which name user want file to save in the database.

Frame: When random images are generated using "Generate" button by user, then to display generated image to user on the screen, frame is used. It is required to display image to user so that user can confirm is generated image is useful for save or not.

➢ OUTPUT

There are two different outputs for this tool one is permanent and other is temporary output. When user generate new random monster image using button "Generate" the output image will display one the screen, it is s temporary output as it will be disappear if exit is pressed or "Generate" button is pressed again. When user generate image and save images using name for that image and by pressing "Save" button, the image will save permanently until not deleted manually.

➢ DATABASE

Local database is used in this tool to save input and output files. Local database is created in the system file of tool, where input images are saved in png format in the folder name as "images" and output files are also saved in png format in the folder name as "Save". "Image" folder contains images of different parts of monsters.

➢ UML DIAGRAM

Unified modelling language (UML) is a approach to visualise a working of system using collection diagram and is considered as standard for modelling software development.

To visualise the working of this tool, Use Case Diagram is intended to capture a system's dynamic aspect. Use case diagrams to collect system requirements including inner and external factors. In Use case diagram, use case are prepared and actors are recognised when a system analysed to obtain its functionality. Following is the use case diagram for "Graphical Random Monster Generation":

VII. IMPLEMENTATION

a) PROGRAMMING LANGUAGE

The Java platform comprises of programming interfaces for Java applications (APIs) and virtual machine Java (JVM). Codes compiled by API libraries can be used in any program. Java is one of the programming languages most commonly adopted, used by some 9 million developers globally and running on 7 billion machines. Java programming language is used popularly to create web applications. It gives flexibility to developer to develop or write code on nay machine and run program on any machine. For numbers of devices, like mobile and computer, java us used to build applications and platform, as well as java is a key language for networking. Java is also used to produce vibrant programs running parallel or embedded in Web pages. Java is object-oriented and class based programming language. “Graphical Random Monster Generator” tool is developed in java as it is flexible to run on any platform securely regardless of operating system and architecture of device, As long as the Java Runtime Environment (JRE) has been installed in the device.

b) SOFTWARE USED

“Graphical Random Monster Generation” tool is developed using Eclipse IDE to write java program and Adobe Photoshop software to edit images of monster used in it.

Eclipse IDE: Eclipse Integrated Development Environment is famous for developing java applications. This tool is developed using latest version of eclipse that is “Eclipse Oxygen”. Codes can be written and program can be built without using IDE where programmer needs to select, integrate, and manage all these tools separately. IDE is used to developed tools as they are integration of different development related tool as a single framework. IDEs are designs to simplify the software development work and also they are useful to reduce the mistakes while writing codes as well as give hint to write some basic code or functions in some cases. An IDE uses single graphical user interface to access an editor, compiler or interpreter and debugger.

Adobe Photoshop Software: It is a raster graphics editor and also popular for digital art. Photoshop can edit and compose various layers of raster images, masking, and several colour models. Photoshop use its own PSD and PSB file format which support these features. For the development of this tool Photoshop CS5 is used. Photoshop is used to edit the existing monster images or data. Using this software, rough graph was defined to determine the size of every part of monster in new generated monsters, also original images were divided into different parts and modified to get fit into defined graph.

c) PROCESS

Graphical Random Monster Generator is a tool developed using Java programming language for writing the algorithms which is required to instruct the tool to perform some task, Eclipse IDE for writing all the Java codes for executing algorithms and Adobe Photoshop software for modifying the original images.

Development of this tool is gone from different phases as define above. It started from research about the procedural content generation and the importance of procedural content generation. From research it was clear that procedural content generation is useful and an effective way to generate content by limits the manual interference. Procedurally generated contents are useful in game design to make game more entertaining with low cost and efforts. It is possible to generate different contents using procedural generation method like generating different levels for a game, random mixing of tiles for a puzzle games, different and random position of enemies in a game, different weapons that can be used in game, different character, many more. Already developed tool of different content generating were the motivation for this tool. As in research, a tool was found which generates random monster description [26] and other was found which generates character with different objects [26]. Markus Friedl [27]in his book Online Game Interactivity Theory discussed about the effect of interactivity in all types of online games like player to player, player to computer etc. He mentioned that it is even challenging for most experienced game developer to design the effective interactivity. A book by Mark J.P. Wolf [24]Building Imaginary World explains that building an imaginary world helps to make audience engaged into the game. Building imaginary world stands for developing different contents. Building different imaginary world need different contends and procedural content generation is the easiest method to generate content.

There are different tools for different character generation similar to them this tool Graphical Random Monster Generator is developed, as monster are also a part of imaginary world and useful in many games. To develop this tool, different sources were used to collect data. Most of the images taken for the development of this tool is from the Monster Manual. This tool uses few monster images as data for new images generation. After discarding many images as they were not fit for the frame or size graph designed for the new random monster generation these images were get selected. This tool is developed to aim to generate thousands of new random monster images.

Every selected image of existing monsters were divided into main three parts using different tools of Photoshop, pen tool to select area to be separated from original images, move tool to move images to make new layer, free transform to stretch and squash the image to set into size standard graph for images. Images were edited at a standard to make it suitable for other monster's parts. Some parts of images were removed using basic rubber tool in Photoshop. Out of three parts, middle part of images was again divided into two parts. It was difficult to merge a part of image with other, as in most of the images some part of a monster body overlaps the other part, so images were again divided into parts and everything was saved into local database.

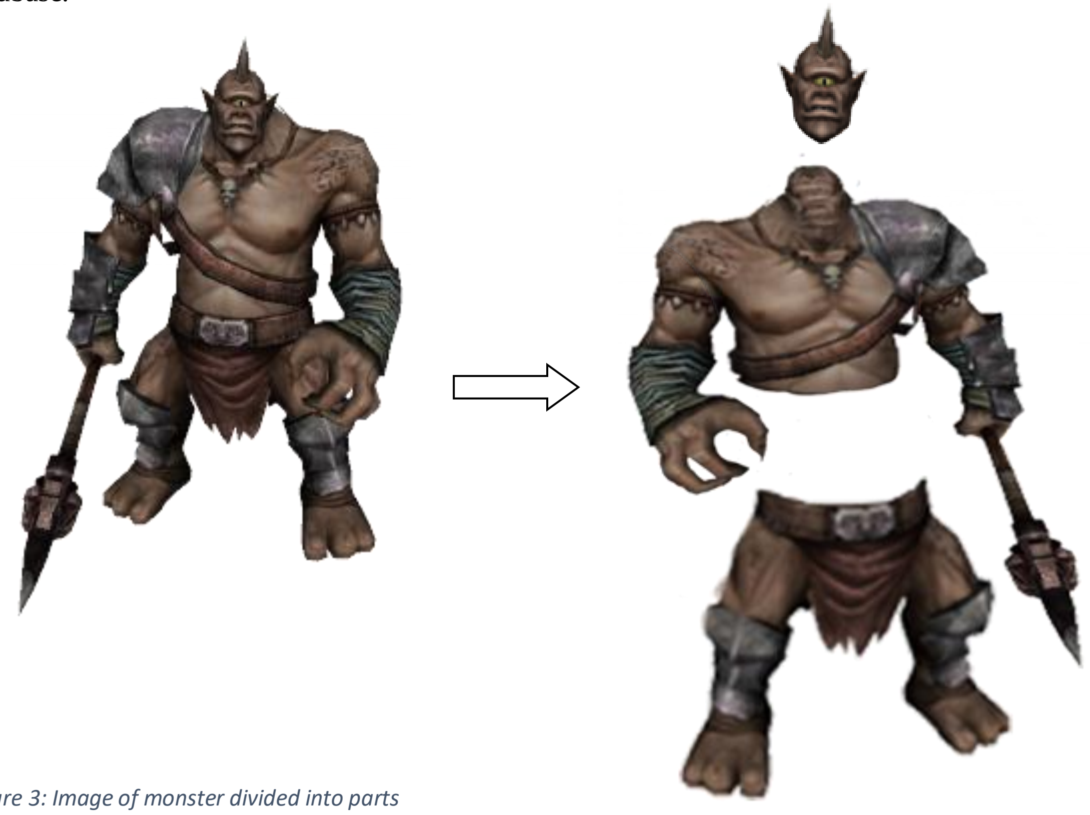


Figure 3: Image of monster divided into parts

After collecting data into database, an algorithm was written to select random numbers and generate new image. For this, Java programming language was used to write code to execute algorithm in Eclipse IDE. It is easy to design frame in java as well as java program are flexible to develop and run on any system with JRE so Java was used to develop this tool. JComponent of Java is used to create JFrame for this tool to make GUI where users can interact with tool easily. In the frame JPanel, button and textbox were added. One of the button names as GENERATE is for generating images. For this, Algorithm was written to generate three random numbers for selecting three basic parts of different monsters, where middle part of which is further divided into two will get selected by using same number generated for the middle part of the body and combine them at the position defined for particular part. After generating image, to display on the screen it was temporally saved into database. As it is temporary, it will get removed from the database on the closing of the tool. To display image on the screen, temporary image is set as ImageIcon. ImageIcon is displayed in JPanel inside the JFrame. And on every time with the click on GENRATE button, new random image will generate and ImageIcon will get update with new temporary image and become new image icon which can be seen on the screen. A text box in the tool within the frame is for writing the name of a monster by which name user want to save that particular image in the database. In this tool, along with GENERATE button there is another button which is named as SAVE button, as name describes this button is used to save image generated with the value given in the text box at the time of clicking on the save button. When clicked on SAVE button, button will first read the text box value and take that value to the save method which define that save method use the text box value to make copy of current generated random image and save into output database. When a user use this tool to generate new monster which display on screen and user like to save that image for further use then user can give some name to that new image using text box and can save permanent into output database from where user can use image later until not deleted manually.

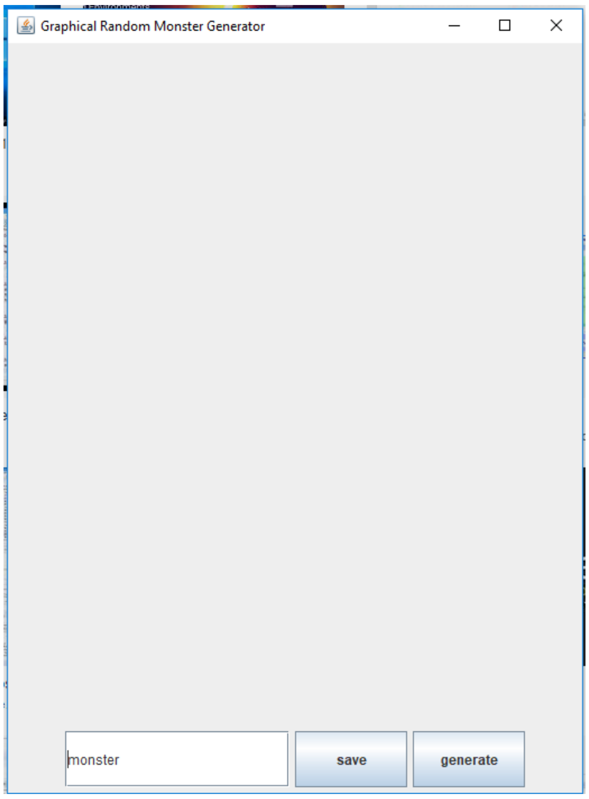


Figure 4: UI (User Interface) of Graphical Random Monster Generator Tool

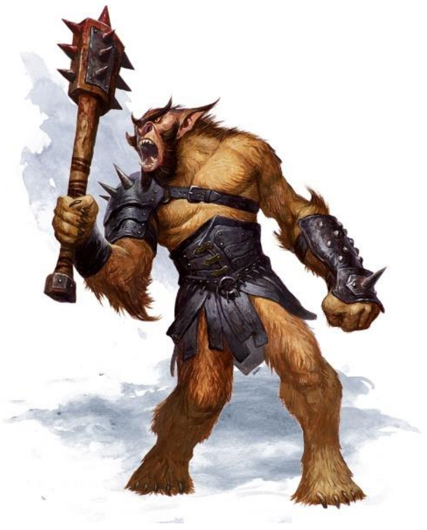
This is the process of developing tool for generating random monsters graphically and the use to generate graphical random monsters from this tool. This is useful to give a basic idea of a monster to games artists which can be used in games with further modifications. For example, A Game Artist can use this tool to get an idea for new monster and can design a 3D monster from this 2D image idea. This tool is not generating image for direct use, it is developed with the purpose of generating image for giving idea to designers to create monsters which save time spent in thinking of monster appearance by giving many monster design ideas.

VIII. CHALLENGES

Development of Graphical Random Monster Generator faced many different challenges in different phases. After overcome all these challenges this tool is developed to give proper output. There are some challenges and the solutions of the problems faced during the development of the tool.

1. First challenge faced was during the collection of data, where the problem was to specify the types of monster can be generated using this tool. There was idea of generating two types of monster: Humanoid and Animal monster. But when they were manually checked then it was not easy to develop both kind of monsters using single algorithm as well as it requires more editing and modification in the original images which was not the part of plan.

Solution: for solving this problem some images of both humanoid and animal monsters images were tested manually, and the conclusion was to use humanoid monster input and output for this tool.



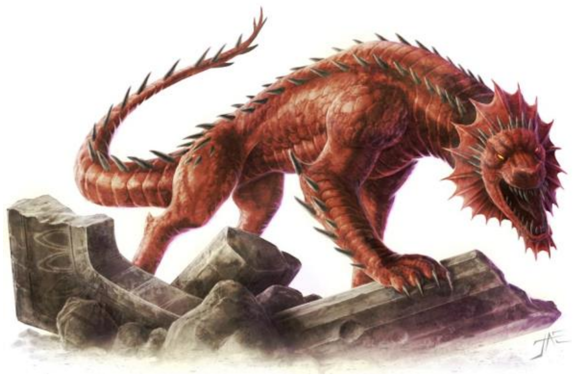


Figure 5: Animal Monster

Figure 6: Humanoid Monster

2. After resolving first issue, there was another challenge of having low resolution images of monsters. As this tool was developed using free available data, it was not easy to find free good resolution images. Some were in good resolution but most of them were not. As there were less data with good resolution, it was difficult to increase resolution for all the low resolution images.
   Solution: as a solution of this problem, instated to increasing the resolution of all images other images were searched with medium resolution. As well as to for remaining images resolution was changed using Adobe Photoshop, low or high according to requirement to make it compatible with other images.
3. After selecting of all images and fixing the size graph for the new images, new challenge arises. There were different parts of different monsters were used in the database. Some monster were thin and tall and some were wide and short, so it was not possible to keep a large head on slim and tall body as it won't fit in the defined size for new monster in the graph.
   Solution: To overcome this problem, solution was found to stretch or squash the parts of monsters while editing according to the given graph and test with different parts of monsters randomly to confirm if they look decent with other monster parts. This solution was implemented on the images as the ratio of the existing monster's body parts does not make any effect as body part images were edited and modified according to new monster generation which is independent of existing monster image size.
4. Data collected for developing this tool was images and they were in 2 Dimension. To make data from raw images, they were divided into different parts of monsters. Generally if it is an image of a humanoid character or other character with different body parts like hand, head, chest, leg, tail, etc., then there are chance to get an image of monster with overlap body parts and this is challenging to get complete body of the same monster in different parts. For example in figure , hand of the

monster cover his some part of leg now where they are divided into different parts, overlapping part can be easily separated but in overlapped part there will be some part missing.
Solution: As collected images are in 2D, there is no other way then completing the incomplete part, to solve this problem. For this, overlapped parts of monster images were painted again to match the character as much as possible. With resolving this issue, some part of monsters where extended to match the other parts of new monster body, as shown in figure.

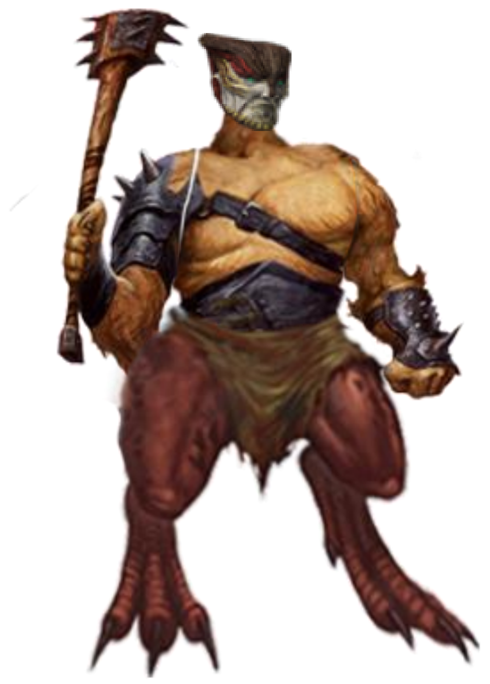


Figure 7: Monster parts are overlapped to make new monster

5. Images where divided in two different parts and randomly connected to other images parts in this tool. In some images, there were parts which were like accessories in the images and creating problem when these parts were used to generate new images. Also need more efforts to edit those parts of image.
   Solution; these parts were not important part and output images are not finalized images which can be used in games directly. It needs more efforts to edit them to fin into the frame. So instead of editing small details, it was better to remove those parts from the images. This does not make any difference in output images as well as reduces the complexity of development of tool.
6. While editing the images for using them as database in the development of this tool, along with all other challenges this was also a challenge to fit the different body parts at correct position with respect to each other. For example, if in new generated image head is at 50px, 100px position and the length of head is 5px so the next part should place at 50px, 105px. This can be resolved by using image processing but that method was time taking and increases the complexity of algorithm.
   Solution; for resolving this problem without increasing the complexity of algorithm the one method used was making graph to mark the position of every part of the image. So that in every new image starting and ending of particular type of parts will be same as defined in the figure.
7. Next challenging point in this process was to prevent images to overlap each other where not required. For example in figure x, mid body of the image is placed in background and the bottom and top parts are placed. But some of the parts of mid body in the image is supposed to be at top not overlapped by bottom part.
   Solution; to overcome this challenge, it was not possible to make bottom part in background as bottom body layer at top position gives proper finishing to the image. So, mid body of monster was

divided into two different parts. Out of which one with chest will remain in background and other with arms and accessories will overlap the bottom body layer.

8. After resolving all problems related to editing now there were problems in algorithm. First problem was that the image generated using random method and combine together by using I/O read method in Java. I/O read method can read an existing file. Using this method algorithm read the existing file from database and combines files by following algorithm. But there is no direct method to display that file on screen with saving it into database.
   Solution; There were no other method except saving it before display, as there is only I/O write method in Java which can be used to write file in system. So to resolve this issue, algorithm is changed to make null.png file of generated monster and save temporarily in the database until program is running. It will get deleted automatically as program exists.
9. Last challenge of this tool development was to display generated image on the screen using JPanle. This was done by set image icon in Java. Here the problem was, when image icon is set it display the image generated but when image is regenerated, is display both next to each other. To solve this, before generating new image, method called to set image icon as NULL, so now only currently generated image will be display. But with the solution of one problem there was another problem that every time image generates and display on the screen, it shift with some pixels.
   Solution; for this problem get resolved by removing JPanel from the tool. As there was a JPlanel in the tool where image icon was set, and by calling generate button for generating image and display, new JPanel was also generating which caused this problem.

IX. RESULT

Development of Graphical Random Monster Generator faced many challenges. After resolving all the issues occur during the development of image editing and algorithm writing, tool is ready to give expected outputs. This tool uses the saved images of monster's parts from database, run an algorithm to select random numbers on the bases of these numbers algorithm selects the images and create new image. New images will be saved in temporary file to display on the screen and will be removed automatically from database on the closing of program. Using given text box and SAVE button, monster can be saved by used defined name. This will be saved in output database as permanent file, which can be deleted manually only.

There are some examples of new random monster images generated using this tool. Which show that tool is successfully working after making all required changes it to so solve problems.

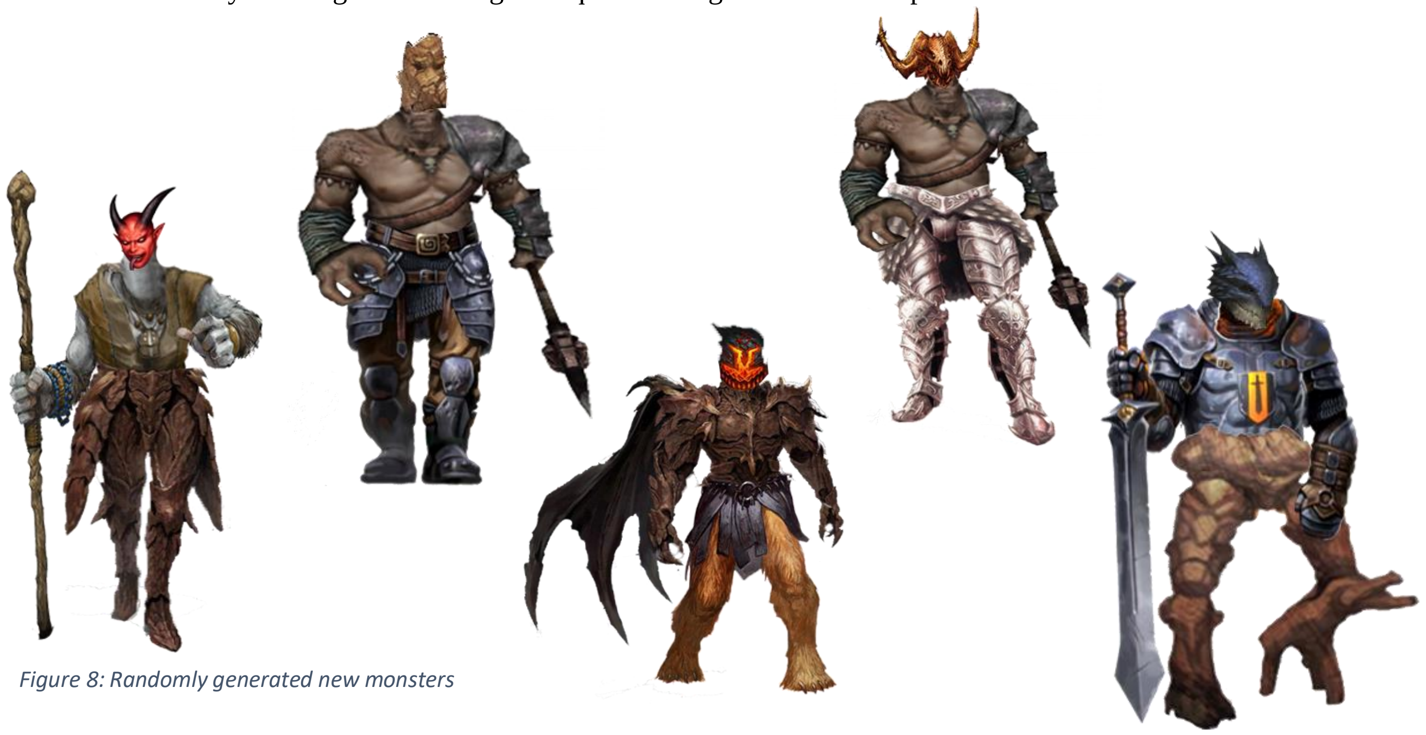


Figure 8: Randomly generated new monsters

X. TESTING

Testing is an important part of any system development. It gives chance to deduct bugs in system before demonstrating to client. There are some tests done while development of this tool which helped to detect bugs in the tool. A table given below is list of testing done in the process of development of this tool.

Test	Result	Outcome
While generating random images using this tool, checking the position of Head of the monster with respect to monster's body	Pass	The Monster's head paced at top-centre of monster's body and body is perfectly overlapped by head
Checking the position of body of monster with respect to monster's leg when generated new monsters	Pass	Monster's body paced at top-centre of monster's leg and leg is perfectly overlapped by body
Try to generate animal type monsters along with humanoid monster using same algorithm by keeping algorithm simple within this tool	Fail	This tool is able to generate only one humanoid monsters by existing algorithm, to generate animal type monster need different algorithm and more modification in data
Write algorithm to concatenate image of different parts of monster body to make new monster by keeping length and width of monster constant	Pass	Algorithm written for concatenate runs properly and concatenate images properly to give new monster image with constant length of image
Algorithm written which try to display concatenated new monster image on the screen within the frame	Fail	There is no function defined in Java programming language to display image directly without saving it to database. There are function like IO read to read file from database and IO write to write file into database
Algorithm to display image on screen is corrected and tested again. Algorithm is modified by make generated new image file temporarily save into database and delete when program closes	Pass	Algorithm run perfectly and display newly generated monster image using IO write on the screen and delete temporary image on program close
Display image on the screen within the frame using set image icon without using JLable	Pass	Image display correctly by using JPanel instead of JLabel. Using JLabel to display new image on screen generates JLabel every time along with generation of new image
GENERATE button is used to generate new image using concatenation algorithm	Pass	New monster Image generates using concatenation algorithm and save temporarily into the database
Generated image saves in the database on click SAVE button	Pass	When press on SAVE button, temporarily generated images save into permanent image using name text given by user
Image save by using text as name for monster given by user in text box	Pass	While saving image permanently using SAVE button, program reads text given by user using text filed and save generated image permanently from temporary
Program give error message if monster name already exists	Pass	If monster name already exists program display ERROR; NAME ALREADY EXIST
Indicate if monster is saved with save button	Pass	When monster saved with given name program display SAVED message on the screen
Refresh the ERROR message	Pass	When GENERATE button pressed, program refresh the ERROR message and delete the ERROR message from screen

XI. FEEDBACK

Graphical Random Monster Generator tool was tested by some designers and other users to receive feedback from them about their experience of using this tool. Some output from the feedback are mentioned below. This pi-chart shows the number of user appreciated the tool and its objective of creating random generated monster for game designers. Another pi-chart shows the percentage of user find this tool useful for designers and suggested to recommend to other designers as well.

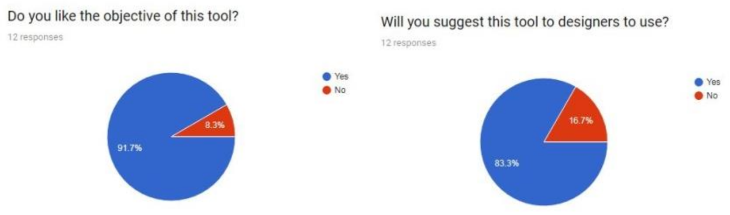


Figure 9: a) Feedback for project objective; b) Feedback for suggesting to designers

Linear chart show the overall rating of the tool from users where mostly user appreciated the tool on the basis of its concept.

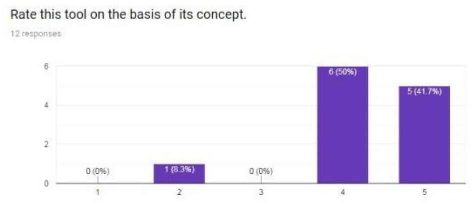


Figure 10: Overall project rating from users

There are some of the details feedback received from users of this tool where they shared their experience with procedural content generated of tool for the first time.

"GRMG is a useful tool to generate multiple characters with interesting features. Though it is randomly generate the output characters are with natural figures. I am intrigued to use this tool for my own purposes in the future"

"This is a really useful tools that can help me increase my productively and creativity in developing new characters for my games and don't have to spend time to much on designing the character. This will help me focus on better improving the game play. Thanks for sharing this tool."

"This is very good tool to developed a game character. I don't need more time to design or to think my characters in game. Thanks for sharing the tools. This is really helpful!"

"This is the one of the best application I have come across so far. Thank you for sharing this useful tool as it gives us a lot of options to select the characters of our choice."

"This tool comes out as a good character generator to me. It helped in automating my task, also I saved my time using it. Improvements can be there such as automatically naming as 1,2,3.. . Overall it helped me in my work"

"Using this tool was fun. Creating different characters randomly saved a lot of my time. The characters are easy and fun to create and the amount of time taken is very less, therefore it increases the efficiency. I can save the ones I like and leave the ones I don't. It was super fun to engage with this tool!"

Overall feedback from the designers and other users manifest that user appreciated the applications concept and work and find the application useful personally.

CONCLUSION

The purpose of this work is to develop a tool to randomly generate graphical monsters using procedural content generation method, which can be useful for character artists or game design artists. Artists can use this tool to get idea for generating new monsters which can be used as reference while designing monster for game. Many researchers did research on procedural content generation method to get procedural contents and mostly are based on game level generation, map generating for game, Artificial Intelligence character generation. This is also based on the research question that is it possible to create plausible monster images while generated randomly using procedural content generation. This tool is designed using Java programming language, Adobe Photoshop and some existing monster's images and splitting them into three sections to randomly match one section of an image with other section of other image and generate an overall new monster images. This research work is done by generating random numbers for developing monsters images. Development of this tool faced many challenges and most of them were successfully resolved and passed almost all the unit tests. Testing complete tool and collecting feedback answer the requesting and manifest that it is possible to create plausible monster images using random method to generate procedural content for games.

FUTURE WORK

At current state use of generated content is still restricted and mostly due to lack of variety of game content available. Therefore, next step of this project involves generating different types of contents using an application. For example more than one type of monster can be generated using the same application or different types of monsters, animals, other characters, buildings etc. can be generated using single tool. Along with adding more variety of contents in application, User Interface of this application needs to be changed. It is possible to add proper save option with selection of folder for output file, open the saved file in the tool instead of open it in photo viewer. Even to make it easier for users to use output it is possible to generate different views of images, which will help user to redraw the character easily.

In Human Computer Interaction HCI design, importance of Procedural Content Generation technology is increasing. Personalisation of user experience, reality time adjustments in content according to the need of user will be helpful to make Procedural Content Generation more effective and meaningful. For example within the random monster generation tool, the length and width of monster or length or width of any particular part of monster can be increased by real time adjustment option in tool. The idea of introducing real time adjustment in tool for designers is similar to the idea of Georgios N. Yannakakis and Julian Togelius, discussed about player real time experience in there paper Experience-Driven Procedural Content Generation [28].

Another concern is about 3D character artist, it is possible to make use of Procedural Content Generation tool easier for them if content can be generated into 3D. For example, tool can generate 3D models of monsters instead of 2D images which can be directly used in 3D games with some modification by modelling artists according to the requirement of game. The concept of making tools with reality time adjustment will work more effectively on 3D models.

REFERENCES

[1] S. M. J. V. D. V. a. A. I. MARK HENDRIKX, “Procedural Content Generation for Games: A Survey,” p. 24, FEBRUARY 2011.

[2] G. Smith, “An Analog History of Procedural Content Generation”.

[3] “Quantum: The Game of Modern Life (Game),” The Quantum Game Company, 1984.

[4] J. K. E. S. D. a. Y. G. Togelius, “What is procedural content generation?: Mario on the borderline,” in *Proceedings of the 2nd International Workshop on Procedural*, (2011).

[5] J. Spann, “ What do you get when you cross a Dungeon Master with a computer? Dragon VII,” pp. 42-48, 1983.

[6] H. WANG, “Proving Theorems by Pattern,” *American Radio History*, vol. XL, 1961.

[7] V. V. a. J. A. Brown, Evolving dungeon crawler levels with relative placement, acm, 2012, pp. 27 -- 35.

[8] S. D. a. J. Togelius, “Procedural Content Generation Using Patterns as Objectives,” p. 325–336, 2014.

[9] M. Gardner, “ Mathematical Games: The fantastic combinations of John Conway’s new solitaire game "life",” *Scientific American*, pp. 223:120–123,, 1970.

[10] G. N. Y. a. J. T. Lawrence Johnson, “Cellular automata for real-time generation of infinite cave levels,” in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames ’10,*, New York, NY, USA,, 2010.

[11] K. Perlin, “An image synthesizer,” *SIGGRAPH Comput. Graph*, pp. 19(3):287–296,, July 1985..

[12] N. S. M. J. .. N. Julian Togelius, Procedural Content Generation in Games, Switzerland: Springer International Publishing, 2016.

[13] S. M. D. P.-L. Xenija Neufeld, “Procedural Level Generation with Answer Set Programming for General Video Game Playing,” in *Computer Science and Electronic Engineering Conference (CEEC)*, 24-25 Sept. 2015.

[14] C. B. a. F. Maire, “Evolutionary Game Design,” *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND A1 IN GAMES*, vol. 7, pp. 1-16, 2016.

[15] F. F. d. V. ,. a. C. C. Miguel Frade, “Breeding Terrains with Genetic Terrain Programming: The Evolution of Terrain Generators,” *International Journal of Computer Games Technology*, vol. 2009, Dec 2009.

[16] R. Z. a. A. Khoo, “Making the Human Care: On Building Engaging Bots,” in *AAAI*, 2002.

[17] J. E. Laird, “It Knows What You're Going To Do: Adding Anticipation to a Quakebot,” 12 2001.

[18] M. Roberts, “Artificial Actors for Real World Environments,” March 2002.

[19] R. J. Firby, “Adaptive Execution in Complex Dynamic Worlds,” 1995.

[20] A. L. J. T. Gabriella Alves Bulhões Barros, “Playing with data: Procedural generation of adventures from open data,” *Proceedings of the International Joint Conference of DiGRA and FDG*, 2016.

[21] J. &. C. J. &. M. D. &. A. C. &. B. J. &. B. P. &. B. S. &. B. G. &. M. B. D. &. C. N. &. C. A. &. D. S. &. D. F. &. E. K. O. &. F. R. Lehman, “The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities,” March 2018.

[22] J. T. Steve Dahlskog, “Procedural Content Generation Using Patterns as Objective,” vol. 8602, April 2014.

[23] Monster Manual, Dungeons & Dargons, september 2014.

[24] M. J. Wolf, Building Imaginary Worlds, Routledge, December 2012.

[25] “1.2 Design Goals of the Java™ Programming Language,” Oracle, January 1, 1999.

[26] [Online]. Available: https://www.fantasynamegenerators.com/humanoid-descriptions.php.

[27] B. Vedrenne, “GLKT,” [Online]. Available: http://www.glkitty.com/pages/rac.html .

[28] M. Friedl, “Online Game Interactivity Theory,” Cengage Learning, 1980.

[29] J. T. Georgios N. Yannakakis, “Experience-Driven Procedural Content Generation,” *IEEE Transactions on Affective Computing* , vol. 2, no. 3, pp. 147-161, 2011.

[30] G. N. Y. a. J. T. Lawrence Johnson, “Cellular automata for real-time generation of infinite cave levels,” in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games, PCGames ’10,*, New York, NY, USA, 2010.

APPENDICES

a) PROJECT CODE

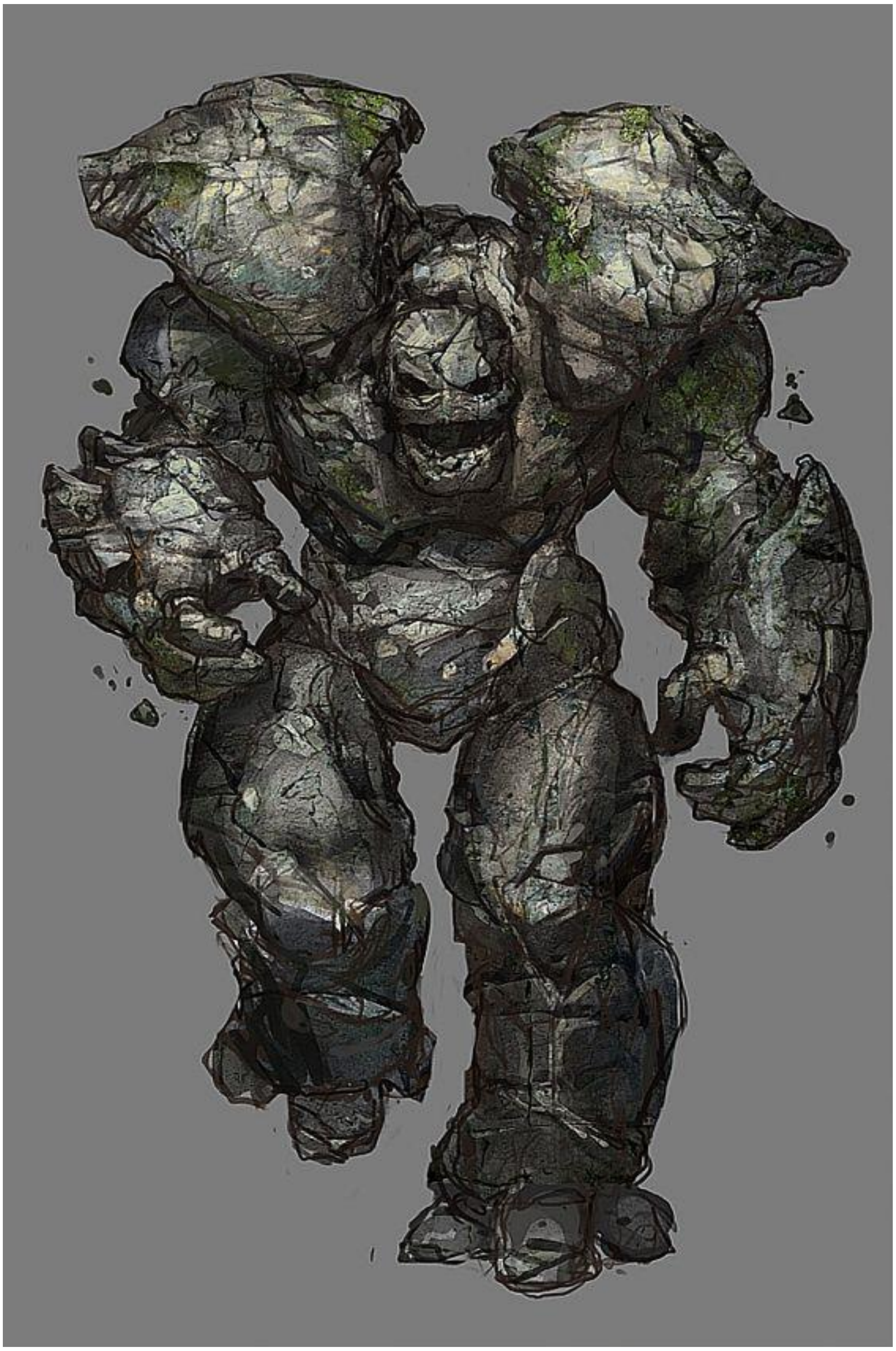
```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.AffineTransform;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.EventObject;
import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class imp extends JFrame{
private JButton button_s;
private JButton button_g;
public JPanel mainPanel;
private BufferedImage imgg;
public String monster1;
public String monster;
private JPanel bottomPanel;
private JLabel lab;
private int n;
private int o;
private int p;
private JLabel l;
private boolean t, er=false;

public imp(){

}
public void panel() {
		mainPanel = new JPanel();
		mainPanel.setLayout(new BorderLayout());
		//mainPanel.setBackground(Color.WHITE);

		bottomPanel = new JPanel();
//		up = new JPanel();
		mainPanel.add(bottomPanel, BorderLayout.SOUTH);
```

```
        //mainPanel.add(lab, BorderLayout.NORTH);
        //up.setBackground(Color.RED);

//      bottomPanel.setBackground(Color.YELLOW);
        button();
        System.out.println("PANEL");
}
public void button() {

        JTextField txt = new JTextField("monster");
        l = new JLabel();
        mainPanel.add(l, BorderLayout.CENTER);
        txt.setPreferredSize(new Dimension(200,50));

        bottomPanel.add(txt);

//      monster = txt.getText();


        button_s=new JButton("save");
        button_s.setPreferredSize(new Dimension(100,50));
        bottomPanel.add(button_s);
        ActionListener listener_s=new ActionListener() {


                @Override
                public void actionPerformed(ActionEvent e) {
                        // TODO Auto-generated method stub

                        monster1 = txt.getText();
                        File f = new File("output\\"+monster1+".png");
                        if(f.exists() ) {
                                if(er==false) {



                                l.setText("ERROR: FILE NAME ALREADY EXIST");
                                er=true;
                                System.out.println("error ");
                                }
                                else {
                                        l.setText("ERROR:PLEASE CHANGE FILE NAME ALREADY EXIST");
                                }

                        }
                        else{
                                        l.setText("SAVED");

                                        if(er==true) {

                                                System.out.println("error cleared");
                                                er=false;
                                        }
```

```
		//	System.out.println("Now Laterst moster is : "+monster1);
			save();
		//	System.out.println(monster1);

			}
		}

	};
	button_s.addActionListener(listener_s);

	button_g=new JButton("generate");
	button_g.setPreferredSize(new Dimension(100,50));
	bottomPanel.add(button_g);
	ActionListener listener_g=new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {
			// TODO Auto-generated method stub

			try {
				//lab.setIcon(null);
				if(t==true) {
					delete();
				}
				image();
				display();



			} catch (IOException e1) {
				// TODO Auto-generated catch block
				e1.printStackTrace();
			}
		}

		private void delete() {
			// TODO Auto-generated method stub

			lab.setIcon(null);
			l.setText(null);
		//	lab.setText(null);

		//	lab.setDisabledIcon(null);
			System.out.println("Delete");
		}
	};
	button_g.addActionListener(listener_g);
}

public void frame() throws IOException {
	panel();
	JFrame frame = new JFrame("Graphical Random Monster Generator");
	frame.getContentPane().add(BorderLayout.CENTER, mainPanel);
```

```
		frame.setSize(960, 1050);
		frame.setVisible(true);
		frame.setDefaultCloseOperation(EXIT_ON_CLOSE);
}

public void display() throws IOException {
		lab = new JLabel();
		mainPanel.add(lab, BorderLayout.CENTER);
		//mainPanel.add(lab);

		BufferedImage image = ImageIO.read(new File("output\\"+monster1+".png"));
		Image newimg = image.getScaledInstance(1000, 1000,  java.awt.Image.SCALE_SMOOTH);
		lab.setIcon(new ImageIcon(newimg));
		//lab.setText(monster1);
		t= true;
		System.out.println("DISPLAY");

}

public void image() throws IOException {
		System.out.println("image again");
		n = (int) (Math.random() * (100/4)+1);
		o = (int) (Math.random() * (100/4)+1);
		p = (int) (Math.random() * (100/4)+1);
		if(n==o || o==p || p==n) {
				System.out.println("GO TO image again");
				image();

		}
		File file1 = new File("images\\"+n+"b.png");
		File file2 = new File("images\\"+o+"a.png");
		File file3 = new File("images\\"+p+"c.png");
		File file22 = new File("images\\"+n+"bb.png");


		System.out.println(n);
		System.out.println(o);
		System.out.println(p);

		BufferedImage img1 = ImageIO.read(file1);
		BufferedImage img2 = ImageIO.read(file2);
		BufferedImage img3 = ImageIO.read(file3);
		BufferedImage img22 = ImageIO.read(file22);

		//int widthImg1 = 500;
		//int heightImg1 = 500;
		int widthImg1 = img1.getWidth();
		int heightImg1 = img1.getHeight();
		System.out.println("widthImg1 "+ widthImg1);
		System.out.println("heightImg1 "+ heightImg1);


		imgg = new BufferedImage(
        widthImg1, // Final image will have width and height as
        heightImg1, // addition of widths and heights of the images we already have
        BufferedImage.TRANSLUCENT);

        imgg.createGraphics().drawImage(img1, 0, 0, null);

        imgg.createGraphics().drawImage(img2, 0, 0, null);

        imgg.createGraphics().drawImage(img3, 0, 0, null);

        imgg.createGraphics().drawImage(img22, 0, 0, null);
        System.out.println("GENERATED");
        //File final_image = new File("C:\\Users\\vandana\\Downloads\\image 3\\"+monster1+".png");
        save();


}

public void save() {

        new imp();

        File final_image = new File("output\\"+monster1+".png"); // “png can also be used here”
        //System.out.println("it should save as.........................: "+monster1);
        monster1=null;
        //System.out.println("it should save as!!!!!!!!!!!!!!!!!!!!!!!!!!!: "+monster1);
        System.out.println("print");
        File final_image2 = new File("output\\null.png");
        final_image2.deleteOnExit();

        try {
                ImageIO.write(imgg, "png", final_image);
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
}


import java.io.IOException;

public class m {
        public static void main(String arg[]) throws IOException{
                imp imp = new imp();
                //imp.image();
                imp.frame();

        }
}
```

b) EXISING MONSTER IMAGES



©2015 Paizo Inc | konstantin vavilov
artvavilov.tumblr.com