

Vilnius gediminas technical university

faculty of fundamental sciences

department of information technologies

Dmytro Teplov

**Bachelor thesis title in lithuanian**

**Procedural generation for diverse applications: creating a versatile 2D world map generator**

Final Bachelor Thesis

Information Technologies Study Programme, State code

Information Technolodies specialization

Study Field of Informatics

Vilnius, 2024

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Fundamentinių mokslų fakultetas

Informacinių technologijų katedra

PATVIRTINTA

Katedros vedėjo

doc. dr. Dmitrij Šešok

Dmytro Teplov

**Bachelor thesis title in lithuanian**

**Procedural generation for diverse applications: creating a versatile 2D world map generator**

Baigiamasis bakalauro darbas

Inžinerinės informatikos studijų programa, valstybinis kodas 6121BX033

? specialization

Informatikos studijų kryptis

|  |  |  |
| --- | --- | --- |
| **Vadovas** |  |  |
|  | (Pedag. vardas, vardas, pavardė) |  |
| **Konsultantas** |  |  |
|  | (Pedag. vardas, vardas, pavardė) |  |

Vilnius, 2024

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

*Vilnius Gediminas Technical University*

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

*Faculty of Fundamental Sciences*

INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

*Department of Information Technologies*

PATVIRTINA / *APROVED*

##### Katedros vedėjas / *Head of the department*

doc. dr. Dmitrij Šešok

**BAIGIAMOJO BAKALAURO DARBO UŽDUOTIS**

***ASSIGNMENT OF BACHELOR THESIS***

Vilnius

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Studentei | | Name Surname | | | |
| *For student* | |
| Baigiamojo darbo (projekto) tema: | | | Title in Lithuanian | | |
| *Bachelor thesis topic:* | | | *Title in English* | | |
|  | | | |  | |
| Baigiamojo darbo užbaigimo terminas | | | | Bakalauro baigiamasis darbas 2 atsiskaitymo data | |
| *The deadline for bachelor thesis* | | | | *Bachelor Thesis 2 exam date* | |
| BAIGIAMOJO DARBO (PROJEKTO) UŽDUOTIS: | | | | | |
| Assignment text in Lithuanian. | | | | | |
| *THE ASSIGNMENT OF BACHELOR THESIS:* | | | | | |
| Assignment text in English. | | | | | |
|  | | | | |  |
| Baigiamojo bakalauro darbo (projekto) konsultantai: | | | | |  |
| *The consultant(s) of bachelor thesis:* | | | | |
|  | | | | | (Pedag. vardas, vardas, pavardė / *Academic title, name, surname* ) |
| Vadovė |  | | |  |  |
| *Supervisor* |
|  | (Parašas / *Signature*) | | |  | (Pedag. vardas, vardas, pavardė / *Academic title, name, surname* ) |

There will be an annotation from mano.vilniustech.lt system (in Lithuanian). You can fill the table of add the one which will be generated in the system

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vilniaus Gedimino technikos universitetas  **Fundamentinių mokslų** fakultetas  **Informacinių technologijų** katedra |  | | | |
|  | | | | |
| **Informacinių technologijų** studijų programos baigiamasis bakalauro/magistro darbas  Pavadinimas **Baigiamojo darbo pavadinimas** | | | | |
| Autorius **Vardas Pavardė** | | Vadovas pedag. vardas **Vardas Pavardė** | | |
|  | | | | |
|  |  | | **Kalba** | |
|  |  | |  | lietuvių |
|  |  | |  | anglų |
|  |  | |  |  |
| **Anotacija** | | | | |
| Baigiamajame magistro darbe nagrinėjamos investicinių projektų vertinimo problemos, vertinimo būtinumas ir tikslai, užsienio šalių patirtis šioje srityje. Išnagrinėti pagrindiniai principai, kuriais remiantis nustatomas investicinių projektų efektyvumas. Pateiktas investicinių projektų efektyvumo vertinimo klasifikavimas, išanalizuoti įvairių jo rūšių skirtumai. Projekto efektyvumas apskaičiuojamas analizuojant investicinio projekto pinigų srautus. Baigiamajame darbe nagrinėjama neapibrėžtumo bei infliacijos įtaka investicinio projekto efektyvumui. Vertinant infliacijos įtaką, išnagrinėtas dabartinių ir prognozuojamų kainų tarpusavio ryšys. Išnagrinėjus teorinius ir praktinius investicinių projektų vertinimo aspektus, pateikiamos baigiamojo darbo išvados ir siūlymai.  Darbą sudaro … dalys: įvadas, ……………………, išvados ir siūlymai, literatūros sąrašas.  Darbo apimtis – 96 p. teksto be priedų, 24 iliustr., 16 lent., 34 bibliografiniai šaltiniai.  Atskirai pridedami darbo priedai. | | | | |
|  |  | |  | |
| **Prasminiai žodžiai:** bent 5 žodžiai | | | | |

There will be an annotation from mano.vgtu.lt system (in English).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vilnius Gediminas Technical University  **Faculty of Fundamental Science**  **Information Technology** department |  | | | |
|  | | | | |
| **Information Technology** bachelor study program final thesis  Title **Title of the thesis** | | | | |
| Author **students** **Name Surname** | | Supervisor prof. dr. **Name Surname** | | |
|  | | | | |
|  |  | | **Language** | |
|  |  | |  | Lithuanian |
|  |  | |  | English |
|  |  | |  |  |
| **Annotation** | | | | |
| Annotation text  … | | | | |
|  |  | |  | |
| **Keywords:** … | | | | |

**Table of Contents**

[Introduction 9](#_heading=h.30j0zll)

[1. Analysis and Overview of Relevant Technologies 11](#_heading=h.1fob9te)

[1.1. Review of Similar Systems 11](#_heading=h.3znysh7)

[2. System Requirements Specification 12](#_heading=h.1t3h5sf)

[2.1. Overview 12](#_heading=h.4d34og8)

[2.2. Overall Description 12](#_heading=h.2s8eyo1)

[2.3. Specific Requirements 12](#_heading=h.3rdcrjn)

[3. System Architecture Description 14](#_heading=h.z337ya)

[3.1. Architecture 14](#_heading=h.3j2qqm3)

[3.2. Dynamic Behavior of Architecture 14](#_heading=h.1y810tw)

[4. System Documentation 15](#_heading=h.4i7ojhp)

[5. System Testing 16](#_heading=h.2xcytpi)

[Conclusions 17](#_heading=h.1ci93xb)

[References 18](#_heading=h.3whwml4)

[Annexes 19](#_heading=h.qsh70q)

[Annex A. Title of the Annex 19](#_heading=h.3as4poj)

[Annex B. Priedo pavadinimas 19](#_heading=h.1pxezwc)

**List of Images**

[**Figure 1.** All images must have a title, defining what is shown in the image 13](#_heading=h.2et92p0)

**List of tables**

[**Table 1.** All tables should have titles as well 13](#_heading=h.tyjcwt)

**List of abbreviations and terms**

|  |  |
| --- | --- |
| **RFID**  **NFC**  **QR Code**  **PICC** | Radio Frequency Identification  Near Field Communication  Quick Response code  Proximity Integrated Circuit Card |
| **USB**  **EEPROM**  **RAM**  **SRAM**  **…** | Universal Serial Bus  Electric erasable and programmable read-only memory  Random Access Memory  Static Random Access Memory  … |
|  |  |
|  |  |
|  |  |

**Introduction**

**Relevance of the thesis.**

In the world of in-game cartography, artists have always held the dominating power. Behind all of the world maps from iconic games like The Witcher 3: Wild Hunt, Red Dead Redemption 2 or Control there is always at least one artist, a team more likely. For small teams of game developers who could not afford to commission an artist for such task, there has always been a need for a tool that would help them achieve great results using minimal input. Additionally, artists working on map creation for video games, or even for board games, could be interested in a tool that would expedite their workflow. This tool could be used for layout prototyping or to create a finished map with their own stylized assets.

**Problem of the thesis**

The primary challenge this thesis aims to address is the creation of a user-friendly desktop application that enables individuals, regardless of their artistic abilities, to design stylized 2D world maps for various projects. These projects can range from board games like Dungeons and Dragons to in-game world maps for indie game studios. A significant feature of the proposed tool is the ability to upload custom assets. This allows users to create maps in any style they desire, making it particularly appealing to indie game studios that typically have at least one artist on the team capable of creating unique assets.

However, developing such a tool presents several challenges. These include ensuring ease of use for non-artists, implementing effective procedural generation algorithms for various map features, and allowing for the seamless integration of custom assets. This thesis will delve into these issues and propose solutions to overcome them.

**Objective of the thesis** – enhance the efficiency and quality of 2D map generation, for a variety of applications such as indie video game development, tabletop games, and literature by developing a procedural 2D map generator application.

**Tasks of the thesis:**

1. Analysis of existing tools for map generation and analysis of algorithms for area approximation, texture blending and uniform distribution.
2. Analysis of multiplatform desktop development technologies.
3. Design 2D map generation tool.
4. Develop and test the prototype.
5. **Map generation algorithms and tools**

Map creation is a historically labor-intensive endeavor. Before everything went digital, creation process involved painstakingly filling out a pieces of paper with depictions of relevant area, and even after the digitalization creating maps involves several steps that require time and thought. They are vital in numerous fields, providing a simplified and easily digestible way to represent spatial information and in some cases position of the user. Maps, be it digital or analog, are created by artists and require considerable amount of time. From that point on, the focus of this paper will be on digital 2D maps created for virtual worlds.

Currently, digital 2D map for virtual worlds creation process involves several steps. Creating initial prototype of a map is a first step and it would include rough blocking, where the artist would outline areas like continents, bodies of water, terrain types or, on the smaller scale, cities and towns, roads, forest areas and so on. Most of the time there are several prototypes carried out to figure out the layout of the map. Next step would be to flash out the artistic style of the map and intricacies regarding that. That would typically include the defining the distinct art style and producing various assets in this art style to eventually place and blend them to final canvas [2].

This paper heavily oriented at gaming industry, since it is a primal producer of virtual worlds that require maps, however the goal of the developed application is to satisfy the needs of any potential end-user requiring digital cartographic solution. A fantasy book writer and a board game developer should also be able to benefit from the developed tool.

Developing a tool that would allow to expedite parts of map creation process is not a new concept. However, this project is primarily geared towards accommodating individuals with varying levels of expertise, ranging from those with no prior experience with painting applications to seasoned, experienced artists.

There are several already existing tools and applications that help to accelerate parts or the whole process, but some are focusing too much on giving control to the user and getting too close to overwhelming them if they want to quickly create a map for their project. Others would restrict the process and make almost everything procedurally and leave only minor changes to the user, like background color change.

This is where this project would come as a solution. It will be appealing to both sides. The core mechanic will be the ability to outline the area types on the canvas by choosing an area in the user interface and paint the areas directly in the application. This 2D map generation tool will allow to convert quick brush strokes into a beautiful stylized map regardless of user skill level. In addition to its user-friendly nature, the application will feature a comprehensive toolset designed to satisfy the needs of professionals as well. This toolset will help professionals to not feel any limitations due to the application's procedural nature, as it enables a level of fine-tuning akin to most popular painting applications.

The average independent game developer would find significant value in integrating this tool into their standard workflow. Doing so will alleviate the pressure of generating distinct maps for each project, especially in due to the time and budget constraints often encountered by smaller studios.

Conversely, larger studios would also derive notable advantages from this tool since it would allow for finer adjustments and the importation of uniquely styled assets, a level of customization made feasible by the presence of dedicated 2D artists within the studio. An important aspect of the project lies in its open-source nature, signifying that individuals with sufficient resources have the capability to adapt the application to better align with their specific requirements.

* 1. **Map generation tools and technologies**
     1. **Inkarnate**

Inkarnate is perhaps the most popular map creation tool. It gained popularity by focusing primarily on board game designers. Dungeons and Dragons, being by far the most widely known, is a fantasy tabletop role-playing game that invites players to create their own characters and embark on imaginary adventures, with one player serving as the Dungeon Master to guide the story and interpret the rules. It utilizes maps in rather a complicated manner: Dungeon masters often use Inkarnate to create detailed maps for their D&D campaigns. These maps can include entire worlds, specific regions, or even individual cities and villages [9].

Inkarnate’s strong suit include:

* + - * Quick asset distribution tool that allows to distribute any type of asset with brush strokes. It also allows to import up to 100 your own assets, but only for paid users.
      * Large asset library, including buildings, mountains, trees and many more.
      * Procedural terrain blending feature allows to paint terrain faster as the user is not concerned with blending between water and terrain, it happens automatically.
      * Road tool is used to create stylized paths on the map.

A screenshot of a computer

Description automatically generated

While this tool may satisfy wide range of customers, all of the more advanced features are locked by a subscription payment. The art style of the map can only be changed by subscribed customers, since the asset import is not available otherwise. The terrain blending settings, unfortunately, are not accessible to free users as well. This limitation renders the entire feature impractical, as the conspicuous brown border cannot be altered.

* + 1. **Watabou Procgen Arcana**

Watabou is another map creation tool, that is specifically designed to provide a generic but at the same time uniquely laid out map with basically no input required from the user. After launching the website user presented with a choice of map type: realm, city, district and village. These all differ in scaling and style for the map to be generated. After launching the generation of a certain map type the tool will fully generate the map and it will appear full screen for the user.

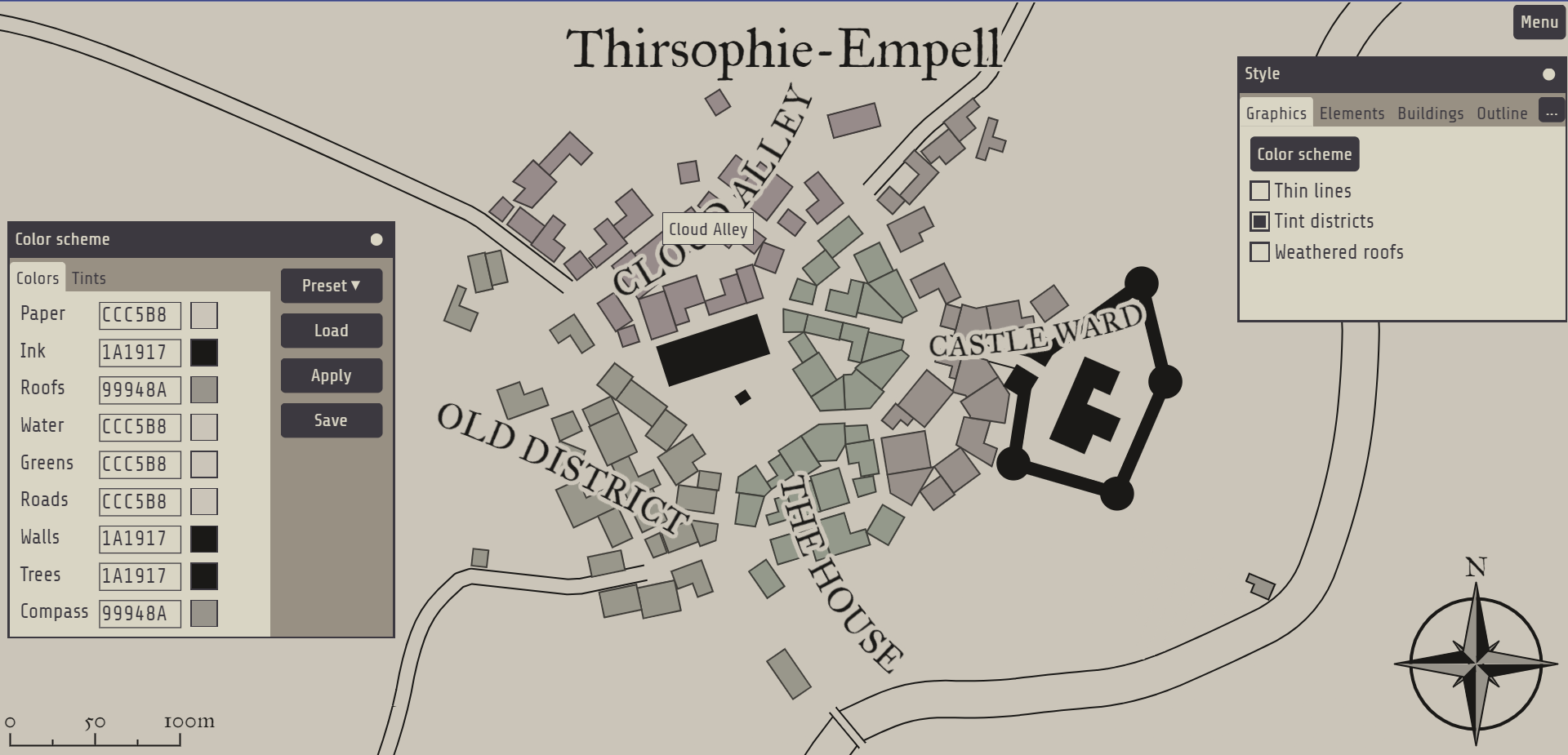
This tool is generating maps in a couple of stages starting with creating randomized graph tree, which will serve as a basis for the map. This graph is meticulously structured using equidistant points, arranged in such a manner that each point is interconnected via an edge to some of its nearest counterparts. This results in a graph that has almost cobweb-like structure, with some randomization. This graph serves as an excellent foundation for modeling towns or continents, cells that have been created by connecting points will be considered as areas and will have a certain type assigned to them. In case of towns, area can be assigned type town district, and hence will be populated by buildings. Other possible areas can be fortress, sea and just land. All of the areas are seamlessly blended together to output fully procedurally generated map.

This approach differs from approach in this paper and allows to create usable map without requiring any input, which is clearly advantageous for users who do not have specific preferences and for whom a generic map would be perfectly suitable.

The adjustments that the user can make to generated map include:

* + - * Adjustments to the graph using different tools like displace, pinch, bloat, rotate and so on.
      * Building arrangement recalculation. Changes the look of a specific area of buildings.
      * Reroll the labels placed on the map or change each label by hand.
      * Change the color scheme.

While these modifications may suffice for those seeking a quick and rudimentary map, they may appear incredibly restrictive to those accustomed to the high degree of flexibility offered by manual map creation. Such users will find graph structure limiting and it does not allow for curved edges and it may become an obstacle when trying to construct more advanced maps. Same goes for color adjustments as this map generator was created with a certain, plain, art style, it doesn’t support gradients and textures.



* + 1. **Competitor 3**
  1. **Algorithms**

Algorithmsare a core part of this project. And as such there will be extensive algorithm analysis to pick the most suited ones.

This section is divided in four parts to discuss algorithms for uniform point distribution over an area, noise generation and texture blending separately. Each section will comprehensively discuss a selection of algorithms, providing an analysis of the advantages and disadvantages associated with each algorithm.

* + 1. **Uniform point distribution algorithms over an area**

Uniform point distribution algorithms are a vital part of the asset distribution during the map creation. Determining the placement of each building within the user-defined ‘town’ area in a manner that avoids overlap and maintains logical coherence presents a significant challenge.

In this section the discussion will be about Poisson-disk sampling and why it is better than basic noise distribution.

Poisson-disk sampling (PDS) is a method that ensures each sample is independent with a certain random distribution, while maintaining a minimum distance from each other [4]. The concept of distance in this context is quite abstract, as the definition of samples vary drastically from application to application. PDS is a method, a strategy, and thus requires an efficient algorithm to execute this method of sampling.

Maximal Poisson-disk sampling (MPDS): a set is considered maximal when no additional samples can be added to the sampling domain without breaching the minimum distance requirement, indicating that the domain is fully occupied [7].

First off, the method itself can be broken down into 5 steps, this unoptimized version is also called “dart throwing” technique [7]:

* + - * Define domain. Setting the boundaries of an area within which the samples should be generated is the first step towards generating the uniform distribution.
      * Choose value that will represent minimum distance between samples.
      * Randomly generate a sample on the domain, and check if it satisfies the minimum distance property. The sample is added to active list, a list containing all valid samples, in case it is located farther than distance r to other and discarded otherwise.
      * To satisfy the maximal condition, step 3 will be executed until there is no place to put an additional sample.

Such method coded naively will perform badly, and in case if maximal condition must be satisfied – extremely bad.

There are two major algorithms for calculating MPDS and each introduce separate optimizations.

* + 1. **Bounding box calculation methods**

Bounding box (BB) is an essential data about a certain area on the screen space or virtual space. It is defined as approximation of an area using arbitrary shapes like box or circle which allows to represent this area with minimal amount of data. In this project it will be used to approximate pixel areas like forest, urban area or castle defined by the user. Since storing all of the pixel positions as area attribute is extremely inefficient, after the user painted the area, BB will be calculated for this area, which will allow to store only four positions of said bounding box.

BB is usually calculated based on minimum and maximum x and y coordinate values present inside the area. Seems very manageable at first, however, it is only the case when only one area is assumed. This project requires to calculate bounding boxes for each separate area that needs to be populated with assets, like urban area, forest area and castle area. In cases of plain asset distribution calculating bounding box that encompasses all regions of certain type should suffice as based on the bounding box area the MPDS will be calculated and points that are outside of painted areas will just be dropped. As for area specific bounding box, the most complex part will be distinguishing separate painted regions that have the same color. Each type of area will be painted on blend mask with its distinct color, but since there can be several regions of same type, the algorithm for determining should be in place.

* + 1. **Procedural noise generation**

Noise is a term used to describe the random or stochastic variations that occur within an expected signal [12]. Procedurally generated noise will serve as the foundational element for the entire map-making project. Every visual aspect presented to the user will be inherently influenced by this foundational noise. Blend mask, which is a raw mess of colors painted by the user, will be distorted heavily by the generated noise image, which will give much more natural output of the landscape.

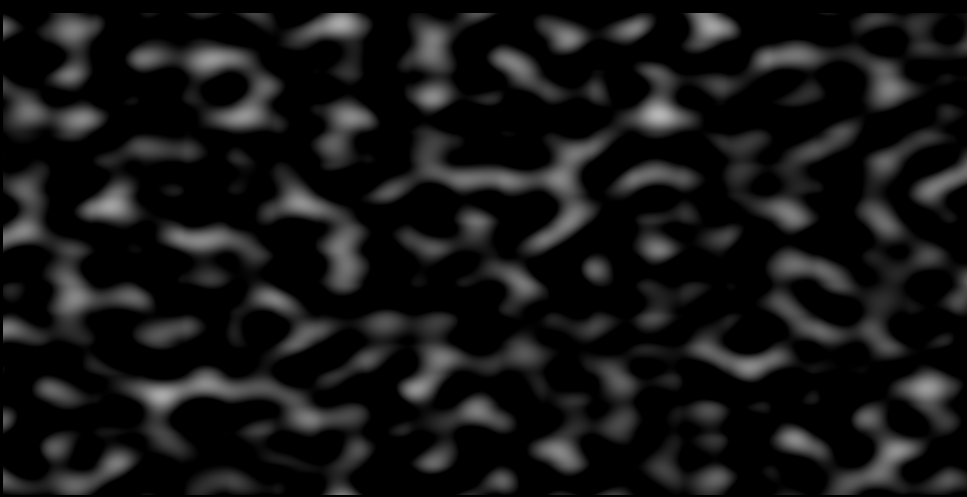
There are several noise calculation methods:

* + - * Perlin noise
      * Simplex noise
      * Value noise
      * Voronoi noise

While value and Voronoi noises have interesting patterns, they have hard white-black characteristic, which is not desirable for terrain generation. Perlin and Simplex on the other hand are both gradient noises, and hence very frequently used for procedurally generating landscapes.

Perlin noise was developed in 1983 by Ken Perling and then published in CIGGRAPH paper in 1985 [13]. After Perlin noise was introduced, it had a significant impact on the field of computer graphics as it allowed to have more realistic imagery. It is widely used still, even after Simplex was introduced by the same author in 2001. Despite its name, simplex noise is more complex in understanding, but it was developed to remedy some shortcomings of Perlin noise, like large computation time in higher dimensions and directional artifacts [14]. Since the gain in computation time is negligible in 2 dimensions, Perlin noise is the choice for this project. The computation of this noise generation method can be broken down into such steps:

* + - * The algorithm starts with a grid of vectors. These vectors can be 1D, 2D, 3D, or even higher dimensions.
      * For each point in the space that we want to compute the noise value of, we take the dot product of the vector from each grid vertex to the point and the corresponding grid vertex’s vector.
      * We interpolate between these dot product values, based on the relative location of the point within the grid cell.

This algorithm will produce such result:  


* + 1. **Wave function collapse**

Wave function collapse method is used widely in procedural generation because of its simplicity and high quality of output results. The method is designed to generate an output based on provided rules, for example it can generate a poem by using provided words and rules like which word rhymes with which. Similarly, it is used to generate any type of imagery, using unique tiles and rules for each tile border telling how selected tile can be attached to other tiles.

This type of generation will be valuable for this project since one feature will heavily rely on it. This feature being castle generation, although due to user’s ability to import custom assets it can be anything else.

Wave function collapse method for this project would have such steps:

* + - * Divide bounding box of pixel area into squares. These will become our tiles. In case of plain bounding box after generating all the squares, some of them might not contain area pixels, in such cases “empty” squares will be discarded.
      * Choose square with least entropy and assign to it a tile randomly chosen from allowed tiles. The lower the number of allowed tiles for a square the lower its entropy. In case of castles, border squares will have least entropy at the start as not all tiles will fit for the border.
      * Reflect the consequences of the choice on all the adjacent squares that not yet been decided.
  1. **Analysis of multiplatform desktop development technologies**
     1. **Programming Language**

The 2D map generation tool, as any other graphically intensive desktop application, consists of 2 major parts: graphics programming interface and programming language used to deploy the commands to graphics card. In this section different possible programming languages will be analyzed to ultimately decide which language to use. This project is focused on real time map creation and hence the key deciding factor will be high computation speed, among others. The programming languages considered for this project are Java, C++, C# and Python.

Java

C++

C#

Python

Adding to that the simple fact that this project in its structure is enormously similar to that of a video game: real time algorithm computations, graphics card utilized for rendering to boost the performance even further and so on.

* + 1. **Graphics Application Programming Interface**

A Graphics Application Programming Interface (API) is a set of instructions that specify, what function should have what output and how it should perform [16]. These instructions are created by owners of the API, and it is up to the graphics card manufacturers to implement these functions. Choosing graphics API that suits best for this project is very important as major part of 2D map generator features will be developed using the certain graphics API. There are several graphics API’s available for building such an application, most popular of which are DirectX, OpenGL, Metal and Vulkan.

Microsoft DirectX was first introduced in 1995 and with its latest version twelve comprises a set of application programming interfaces (APIs) designed to manage various multimedia-related tasks, with a primary focus on game programming and video applications, specifically tailored for Microsoft platforms. While it is usually the best choice for developing Win32 applications, the platform on which map generator will be developed, it does not satisfy the key requirement of multiplatform support.

OpenGL (Open Graphics Library) is another graphics API, it is developed by Kronos Group and released in 1992. From that point on, OpenGL became extremely popular among graphics developers and received several version updates. Main strengths of OpenGL nowadays include large community of developers, ease of use, built-in shader compiler and more. OpenGL is suitable for this project as it allows for multiplatform development with considerably easier setup and development process compared to others.

Metal, an API developed by Apple, is a low-level, low-overhead solution for hardware-accelerated 3D graphics and compute shaders. It made its debut with the release of iOS 8 in 2014 and have since then been used in iPhone, iPad and their computers. With recent addition of ray tracing support, Metal is becoming very powerful, allowing developers to fully utilize Apple’s diverse silicone portfolio. As advanced as Metal API is, it is not suitable for this project for the same reason as DirectX – it is a single platform solution.

Vulkan is an API also developed by Kronos Group and introduced in 2016 as a substitution for OpenGL. Vulkan was created to provide better support for new generations of graphics cards and give even more low-level interface for developers. This approach of Kronos Group ensured that Vulkan was never meant to be easy in its use, but with more complicated approach it can be faster in some cases than OpenGL. A lot of processes that were automated in OpenGL were instead left for developer to implement them. As its predecessor, Vulkan as well multiplatform, and hence is a valid choice for this project.

The choice between Vulkan and OpenGL was ultimately influenced by several factors. These included the more accessible learning curve of OpenGL, bolstered by a substantially larger pool of learning resources. Additionally, the deployment process for OpenGL was significantly simpler. While Vulkan may offer faster performance, the difference was not so revolutionary as to outweigh these considerations.

1. **System Requirements Specification** 
   1. **Diagrams**
2. **System Development**

**Conclusions**

1. …
2. …

**References**

1. T. Gao, J. Zhang and Q. Mi, Procedural Generation of Game Levels and Maps: A Review, 2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Korea, Republic of, 2022, pp. 050-055, doi: 10.1109/ICAIIC54071.2022.9722624.
2. Röhl, Tobias & Herbrik, Regine. (2008). Mapping the Imaginary - Maps in Fantasy Role-Playing Games. First publ. in: Forum Qualitative Sozialforschung 9 (2008), 3, Art. 25. 9.
3. Yao, X., Zhang, Y., Bao, F. et al. The blending interpolation algorithm based on image features. Multimed Tools Appl 77, 1971–1995 (2018). <https://doi.org/10.1007/s11042-017-4379-5>
4. Wang, T. (2021). Poisson-Disk Sampling: Theory and Applications. In: Lee, N. (eds) Encyclopedia of Computer Graphics and Games. Springer, Cham. <https://doi.org/10.1007/978-3-319-08234-9_398-1>
5. Yan, DM., Guo, JW., Wang, B. et al. A Survey of Blue-Noise Sampling and Its Applications. J. Comput. Sci. Technol. 30, 439–452 (2015). <https://doi.org/10.1007/s11390-015-1535-0>
6. R. Bridson. Fast Poisson Disk Sampling in Arbitrary Dimensions, SIGGRAPH 2007
7. David Cline, Kenric B. White, Parris Egbert, "Poisson Disk Point Sets by Hierarchical Dart Throwing", Symposium on Interactive Ray Tracing, pp. 129-132, Sept. 27.
8. Quan, W., Yan, D-M., Guo, J., Meng, W., & Zhang, X. (n.d.). Maximal Poisson-disk Sampling via Sampling Radius Optimization. NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China.
9. ERGÜN A., Examination of the usability of digital maps according to the geography course curriculum. <https://doi.org/10.32003/igge.1301845>
10. Medieval fantasy city generator by Watabou. itch.io. (n.d.). <https://watabou.itch.io/medieval-fantasy-city-generator>
11. Zand, M., Etemad, A., & Greenspan, M. (2022). Oriented Bounding Boxes for Small and Freely Rotated Objects. IEEE Transactions on Geoscience and Remote Sensing, 60. <https://doi.org/10.1109/TGRS.2021.3076050>
12. Abraham, S. (2023). Noise Functions. University of Texas. <https://www.cs.utexas.edu/~theshark/courses/cs354/lectures/cs354-21.pdf>
13. Perlin, K. (1985). IMAGE SYNTHESIZER. Computer Graphics (ACM), 19(3), 287–296. <https://doi.org/10.1145/325165.325247>
14. Gustavson, S. (2005). Simplex noise demystified. Linköping University, Schweden, 17. Retrieved from <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Simplex+noise+demystified#0>
15. Naim, R., Nizam, M. F., Hanamasagar, S., Noureddine, J., & Miladinova, M. (2010). Comparative Studies of 10 Programming Languages within 10 Diverse Criteria. Citeulikeorg (p. 126). Retrieved from <http://arxiv.org/abs/1009.0305>
16. Vries, J. de. (2020). Learn OpenGL. Learn OpenGL - Graphics Programming (Vol. 42, pp. 194–196).

**Annexes**

* + - 1. **Title of the Annex**

Pagrindinis tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas.

* + - 1. **Priedo pavadinimas**

Pagrindinis tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas tekstas.