# Landscape generation

Hi, my name is A.P. (I’m student of the faculty of the Computer software and information technologies), and the topic of my project is the generation of a 3D landscape.

# Goal

The goal of my work is to develop a program for generating and visualizing the landscape. And you can see the tasks on the slide.

# Representation of landscape data

There are several basic principles of data representation for storing information about landscapes:

* the first one is a heights map;
* the second one is an irregular grid;
* the third one is segmented heights map

# Choosing a generating algorithm

On the slide you can see a comparison of the landscape generation algorithms I analyzed. I chose the diamond-square algorithm.

In this algorithm, the input is a flat surface, the height of the vertices of which is equal to zero. Then values are assigned to the corner heights. After that, the algorithm can be divided into two steps:

* step diamond – it calculates the midpoint of the current square by averaging the values of the corner vertices and adding a special random number;
* square step - it calculates the midpoints of the edges for the current square by averaging the vertices on the left, right, top and bottom and adding a special random number. If one of the vertices goes beyond the border of the heights map, then such a point can either be ignored or considered equal to zero, so that it will decrease closer to the border of the landscape.

Afterwards, the heights map is divided into 4 smaller squares and the diamond, square steps are repeated for them until the squares become points.

# Choosing a hidden-surface removal algorithm

The next slide presents a comparison of the algorithms for removing invisible surfaces that I analyzed. To solve this problem, I chose an algorithm that uses Z-buffer.

This algorithm uses two buffers: a framebuffer and a Z-buffer. The frame buffer is used to store the attributes (e.g. intensity) of each pixel in image space, and the z-buffer is a depth buffer used to store the z-coordinate of each visible pixel in image space. During operation z-value of each new pixel is compared to the depth of the pixel that is already in z-buffer. If the new pixel is ahead of the pixel in the framebuffer, then the new pixel is placed in that buffer, and the z-buffer is also updated with the new z-value. If the comparison gives another result, then no action is taken.

# Main tools used for implementation and research

1. C++ programming language.
2. Cross-platform framework Qt.
3. Development environment Qt creator.
4. Version control system git.
5. Clock() library.

# Program output example

On the next slides you can see examples of the output of my program.

# ZBuffer with threads

I also analysed the execution time of the ZBuffer algorithm without and using parallel threads. The results of which you can see in the table.

As expected, the fastest is the algorithm that uses threads.

# Conclusion

As a result of the work done, the goal was achieved - a program for generating and visualizing a 3D landscape was developed. Also, all the tasks listed on the screen were solved.