Márton Áron Főgimnázium

Csíkszereda

Szakvizsga Dolgozat

Térkép generálás

Irányító tanár:

Csomós Róbert

Diák:

Tankó-Gábor Tamás

-2021-

Tartalomjegyzék

Bevezetés

A térképészet vagy kartográfia fontos része a történelmünknek. Hagyományos eszközökkel, általában papírra rajzolták, később nyomtatták. Az ősember is rajzolt tájakat, de nem tájékozódási célból. Az 1990-es évek elején a számítástechnika óriási változásokat hozott, azóta a térképek nagy százaléka számítógépen készül.

A programom a modern számítógépek potenciálját használja ki, hogy mondhatni véletlenszerű térképet hozzon létre, és rajzoljon is ki. A diamond-square algoritmus segítségével ez a véletlenszerű alkotás bizonyos rendszer szerint mégis összefüggő, valószerű és a Simple DirectMedia Layer által nyújtott lehetőségekkel jól szemléltethető lesz.

Számtalan játékban szükség van egy térképre, és sokszor előfordul, hogy ez a bizonyos térkép minden alkalommal más kellene legyen. Ebben segíthet a program, ami személyre szabható több szempontból is. Amennyiben csak egy térképre van szükség, akkor is segíthet ez az alkalmazás, mivel megkönnyíti és jelentősen rövidebbé teszi a kartográfiát.

Ezentúl, nem csak játékoknál segíthet a programom, fantázia világok térképeinek kitalálásában nagy előrehaladást jelenthet. Ami hasznára lehet úgy könyv íróknak mint sorozat- vagy film készítőknek, hogy a rendelkezzenek egy összefüggő világgal, amit akár ki is lehet rajzoltatni a könyvbe vagy a képernyőre.

Elméleti alapok

1. C++

A programot C++ nyelvben írtam, tökéletesen alkalmas erre a feladatra. A C++ egy általános célú, magas szintű programozási nyelv. Támogatja a procedurális, az objektumorientált és a generikus programozást, valamint adatabsztarkciót. Napjainkban szinte minden operációs rendszer alá létezik C++ fordító. A nyelv a C programozási nyelv hatékonyságának megőrzése mellett törekszik a könnyebben megírható, karbantartható és újrahasznosítható kód írására, ez azonban sok kompromisszummal jár, erre utal, hogy általánosan elterjedt a mid-level minősítése is, bár szigorú értelemben véve egyértelműen magas szintű.

Érdekesség: Mire a nyelvet szabványosították, már rengeteg C++ nyelvű kód készült, került használatba. Mivel a szabvány fejállományok némileg eltértek az eddigiektől, a bizottság érdekes megoldást választott a kompatibilitás megtartására. A régi C++ fejállományok (pl. „iostream.h”) továbbra is használhatóak, de tartalmuk nincs benne a standard névtérben. Az új hivatalos fejállományok („iostream”) szinte megegyeznek a régiekkel, de tartalmuk a standard névtérben szerepel. A szabván C fejállományok (pl. „stdio.h”) továbbra is támogatottak, de tartalmuk a globális névtérben van. A C könyvtárak átvétele szintén a .h eltávolításával történt, beszúrva egy c-t a nevük elé (pl. „stdio.h”-ból „cstdio” lett). Tartalmuk a standard névtérben szerepel.

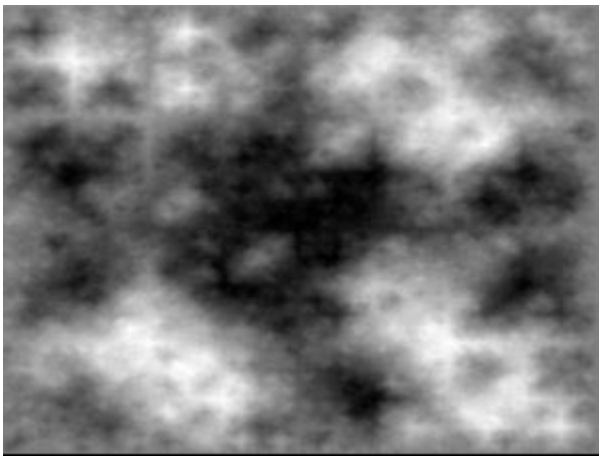
A nyelv bizonyos összetevőire (változók, konstansok, függvények stb.) névvel hivatkozunk. A legtöbb fordító csak az első 32 karaktert veszi figyelembe a nevekben. A név első karaktere betű vagy aláhúzásjel (_) lehet, ettől kezdődően már számok is szerepelhetnek benne. Lehetőleg saját névként ne adjunk meg aláhúzásjellel (_) kezdődő nevet, mert ezek a fordító számára vannak fenntartva (pl. __DATE__, __cplusplus, _MSC_VER). A C++ különbséget tesz a kis- és nagybetűk között (case-sensitive). Az alma név nem ugyanaz, mint az Alma név.

Ezek a kifejezések a nyelv részei, önmagukban nem használhatóak névként.

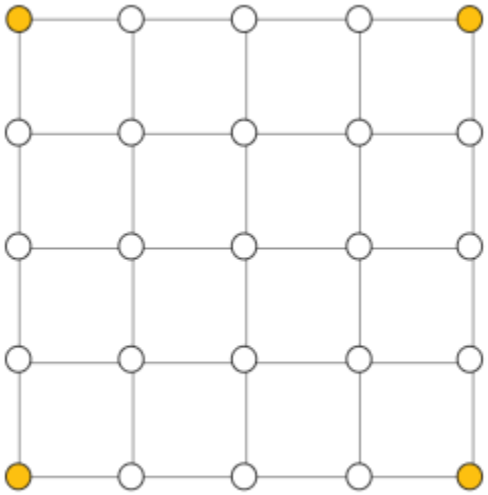
asm	delete	goto	reinterpret_cast	try
auto	do	if	return	typedef
bool	double	inline	short	typeid
break	dynamic_cast	int	signed	typename
case	else	long	sizeof	union
catch	enum	mutable	static	unsigned
char	explicit	namespace	static_cast	using
class	export	new	struct	virtual
const	extern	operator	switch	void
const_cast	false	private	template	volatile
continue	float	protected	this	wchar_t
decltype	for	public	throw	while
default	friend	register	true	

2. Diamond-square algoritmus

A diamond-square algoritmus egy módszer, hogy térképeket generáljunk számítógép segítségével. Más elnevezései a random midpoint displacement fractal, a cloud fractal és a plasma fractal, amiért fraktálokat lehet készíteni a módszer alkalmazásával. Az ötlet először, 1982ben Fourmier, Fussel és Carpenter által volt bemutatva a SIGGRAPHban.



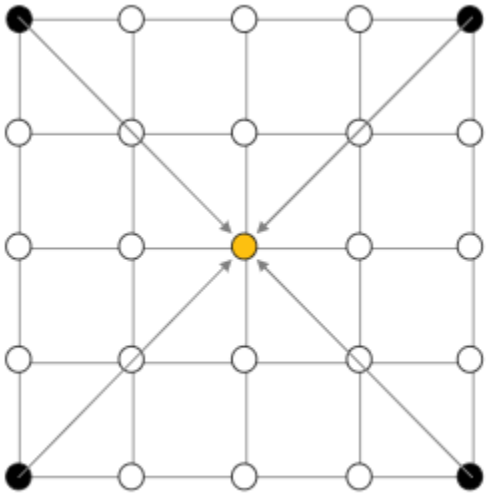
Az algoritmus egy két dimenziós négyzet tömbbel kezd, aminek a magassága és szélessége 2^n+1. A négy sarok pontot kezdeti értékre kell állítani.



Initialize corner values

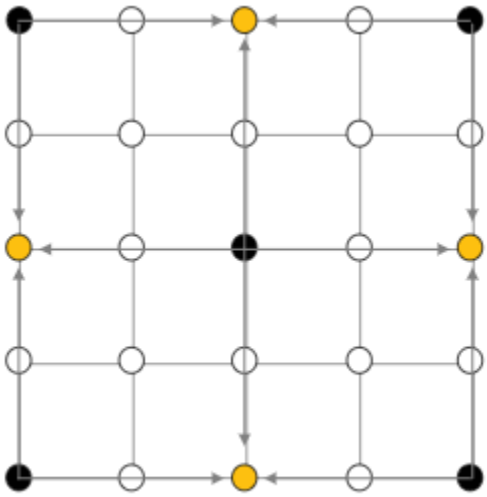
Ez után a diamond és square lépések sorra következnek amíg az összes mező értéket nem kap.

- Diamond lépés alatt minden négyzetnek a tömbből a középpontját egyenlővé tesszük a négyzet négy sarok értékének átlaga plusz egy véletlenszerű értékkel.



Perform diamond step

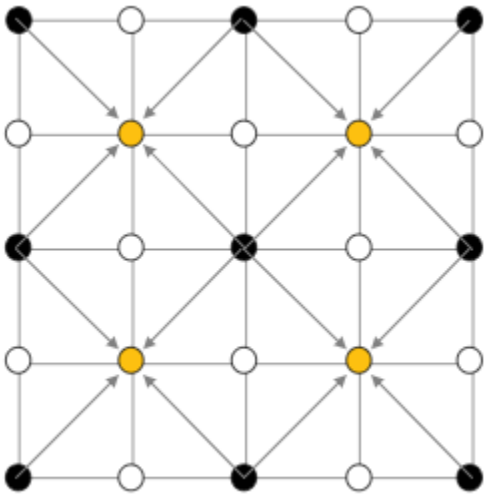
- A square lépés során minden gyémántnak a tömbből a középpontját egyenlővé tesszük a gyémánt négy sarok értékének átlaga plusz egy véletlenszerű értékkel.



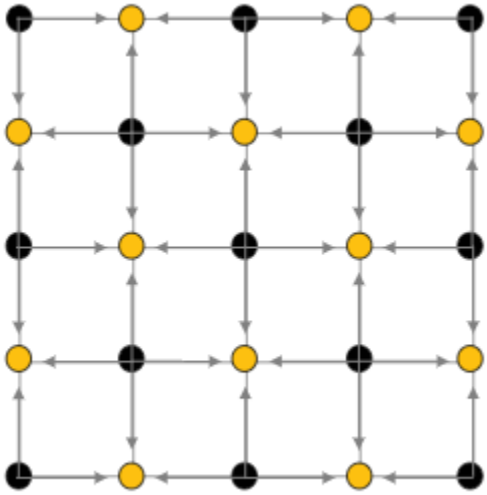
Perform square step

Minden érték adásnál a véletlenszerű terjedelmét csökkenteni kell.

A visszalévő két lépés ábrázolása az 5 x 5-ös tömbben:



Perform diamond step



Perform square step

3. Simple DirectMedia Layer (SDL)

A programom külön ablakban történő kirajzolásához SDL-t használtam, ami egy „multiplatformos” ingyenes, nyílt forráskódú multimédiás szoftverkönyvtár, amely C környezetben íródott, és különböző rendszerek grafikai, hang- és bemeneti API-jai – alkalmazásprogramozási interfészei – felett vett hidat létesít, ezáltal lehetővé téve a programfejlesztők számára, hogy programjaikat – számítógépes játékokat vagy egyéb multimédiás alkalmazást – csak egyszer írják meg, ám azt több platformra is változtatás nélkül lefordíthassák, és ezután a programot valamennyi rendszeren futtassák.

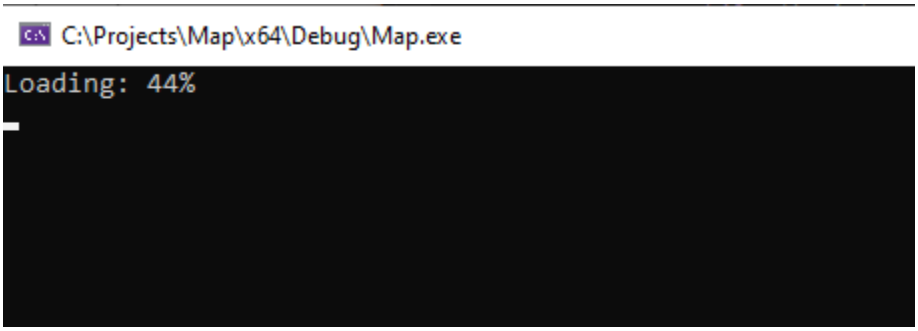
Az SDL nevében benne van a layer, réteg szó, mivel ez egy keret, egy „réteg” az egyes rendszerek különleges tulajdonságai köré. A fő feladata az SDL-nek, hogy egy közös keretrendszert biztosítson ezeknek a tulajdonságok eléréséhez.

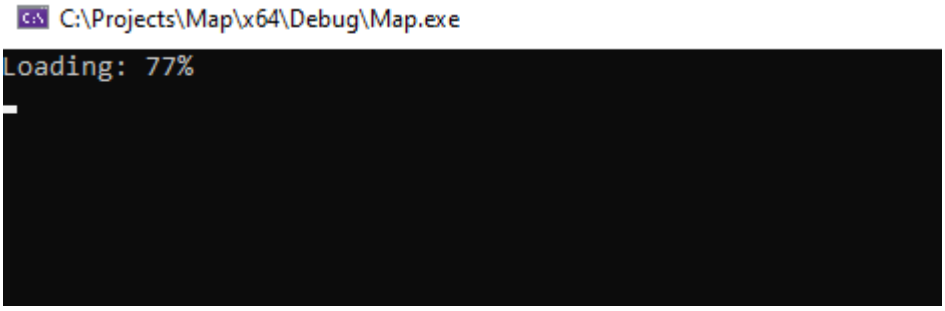
Az SDL szintaktikája függvény alapú; minden művelet az SDL segítségével, függvényeknek adott paraméterek segítségével történik. Kiemelt struktúrákat is használ, hogy az SDL-nek kezeléséhez szükséges információkat tároljon. Van néhány különböző alrendszer, amely függvényeit, az SDL kategóriákba sorol:

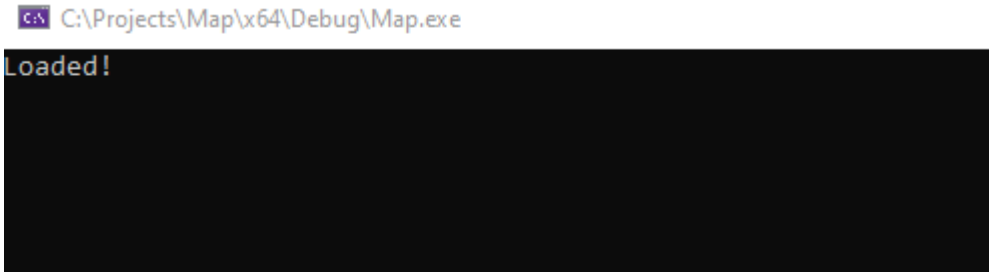
- A Videó, események és folyamatok alrendszere - ez támogatja a videó, több folyamat és események kezelését.
- Az Audió alrendszer - ez a hang működését szolgálja.
- Az Időzítő alrendszer
- A Botkormány alrendszer
- A CD-ROM alrendszer

Felhasználói dokumentáció

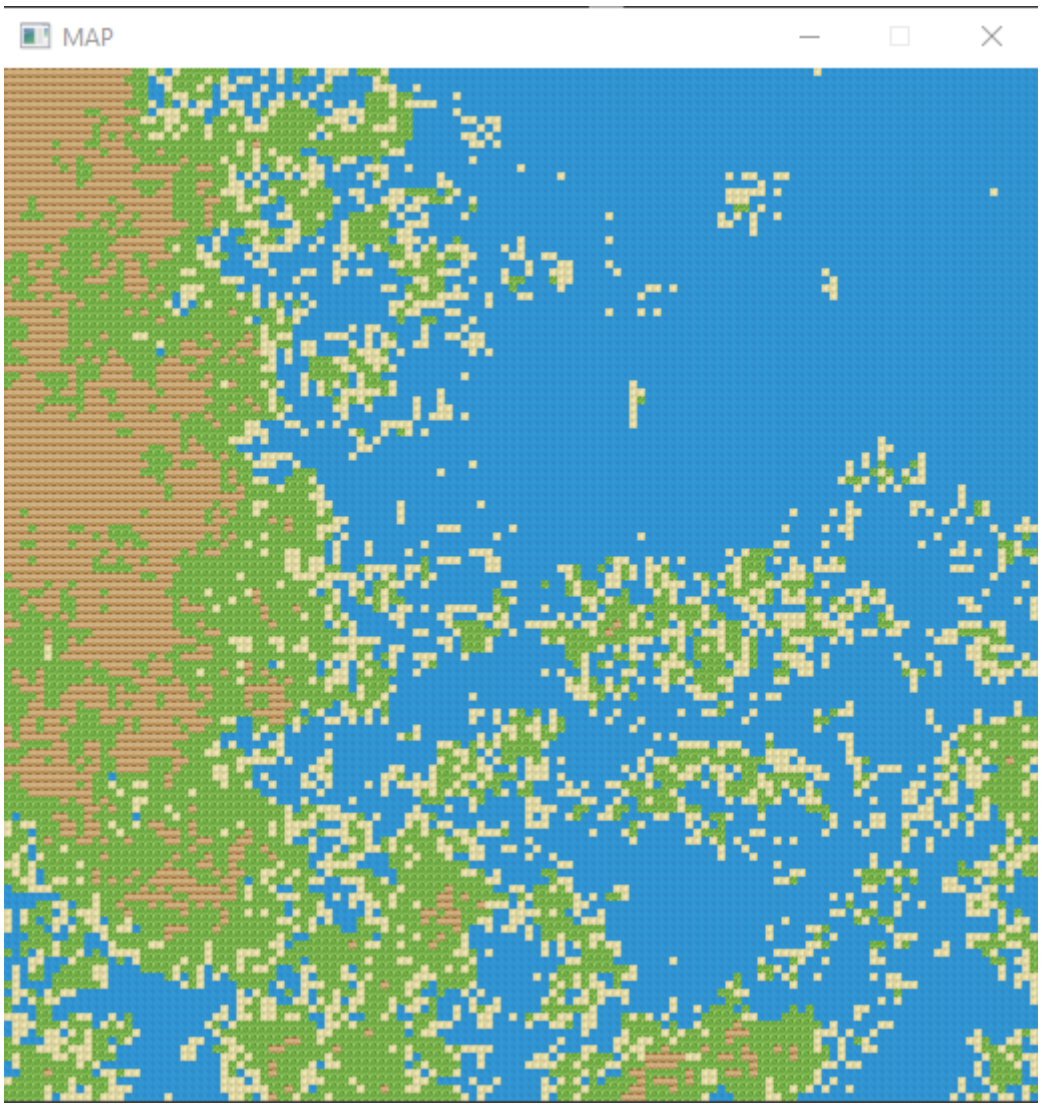
A program előre beállított méretek szerint készíti a térképet, de ez személyre szabható a kódban. Fontos, hogy az algoritmus által igényelt 2^n+1-es méretet adjunk meg. A futtatás gomb megnyomása után a konzol ablakban kiírja a betöltés folyamatát, ami egy modern számítógépen ugyan nagyon gyors, de egy régebbi hardverekkel rendelkezőnél egy kis időbe telhet.







Miután betöltötte a térképet, SDL segítségével megnyit egy új ablakot, amiben kirajzolja a generált térképet. Ennek mérete, ahogy fent említve volt, változtatható. Emellett az is változtatható, hogy milyen arányban legyen víz és szárazföld, erre a későbbiekben kitérünk.



Fejlesztői dokumentáció

A program alapja egy két dimenziós tömb, amely eredetileg nullásokkal van feltöltve. A diamond-square algoritmus igényei szerint a tömb négy sarkának külön kezdeti értékeket adunk.

```
93      float** terr = new float* [SIZE];
94      for (int i = 0; i < SIZE; i++)
95          terr[i] = new float[SIZE];
96      srand(time(NULL));
97      float randomness = 0.75; float negRandomness = -0.75;
98      for (int i = 0; i < SIZE; i++)
99      {
100         for (int j = 0; j < SIZE; j++)
101         {
102             terr[i][j] = 0.0;
103         }
104
105     }
106     terr[0][0] = round((((float(rand()) / float(RAND_MAX)) * (randomness - negRandomness)) + negRandomness) * 100) / 100;
107     terr[SIZE - 1][SIZE - 1] = round((((float(rand()) / float(RAND_MAX)) * (randomness - negRandomness)) + negRandomness) * 100) / 100;
108     terr[0][SIZE - 1] = round((((float(rand()) / float(RAND_MAX)) * (randomness - negRandomness)) + negRandomness) * 100) / 100;
109     terr[SIZE - 1][0] = round((((float(rand()) / float(RAND_MAX)) * (randomness - negRandomness)) + negRandomness) * 100) / 100;
```

Ezután kezdetét veszi a tömb feltöltése értékekkel. Egy for function-ben minden mezőn átmenve és minden esetet feltételezve, kivitelezve ez megtörténik. Ehhez a fent leírt diamond-square algoritmus személyre szabott változatát használja a program. Példaként egy diamond lépés implementálása a kódban:

```
{
    float average =
        round(((terr[i][j] + terr[i + currSize][j] + terr[i + currSize][j + currSize] + terr[i][j + currSize]) / 4) * 100) / 100;
    terr[int(i + currSize / 2)][int(j + currSize / 2)] =
        average + round((((float(rand()) / float(RAND_MAX)) * (randomness - negRandomness)) + negRandomness) * 100) / 100;
    if (terr[i][j]<0.01 && terr[i][j] > -0.01)
        terr[i][j] = 0.0;
}
```

A térkép legenerálása után következik az SDL elindítása, majd a képek betöltése, amelyeket a program kirajzol víz, homok, alföld és hegy mezőkként.

```
206         SDL_Texture* water = loadTexture("water.png");
207         SDL_Texture* sand = loadTexture("sand.png");
208         SDL_Texture* grass = loadTexture("grass.png");
209         SDL_Texture* mountain = loadTexture("mountain.png");
```

A program addig jeleníti meg a térképet, amíg a felhasználó ki nem lép az „X” használatával a MAP ablakból.

```
210         while (!quit)
211         {
212             while (SDL_PollEvent(&e) != 0)
213             {
214                 if (e.type == SDL_QUIT)
215                 {
216                     quit = true;
217                 }
218             }
```

A tömb fel van töltve, minden készen áll, már csak a kirajzolás van hátra, egyszerűen a tömb értékei alapján kiválasztjuk, hogy melyik érték intervallumokra milyen képet rajzoljon ki. Ez személyre szabható, be lehet mindig állítani, a szárazföld és a víz előfordulási arányát az értékek alapján. Példaként a víz mező kirajzolásának kódja:

```
if (terr[i][j] < -0.2)
{
    SDL_Rect block;
    block.x = i * 4;
    block.y = j * 4;
    block.w = 4;
    block.h = 4;
    SDL_RenderSetViewport(gRenderer, &block);
    SDL_RenderCopy(gRenderer, water, NULL, NULL);
}
```

Miután a kirajzolás sikeres volt, a felépített texturákat töröljük.

```
272         SDL_DestroyTexture(water);
273         water = NULL;
274         SDL_DestroyTexture(sand);
275         sand = NULL;
276         SDL_DestroyTexture(grass);
277         grass = NULL;
278         SDL_DestroyTexture(mountain);
279         mountain = NULL;
```

Megmaradt a felépített ablak és a renderelő rendszer, ezeket is kitöröljük, és bezárjuk az SDL könyvtárakat.

```
81  void close()
82  {
83      SDL_DestroyWindow(gWindow);
84      SDL_DestroyRenderer(gRenderer);
85      gWindow = NULL;
86      gRenderer = NULL;
87      SDL_Quit();
88      IMG_Quit();
89  }
```

Fejlesztési lehetőségek

Egy hasznos és nagy lépés lenne két dimenzió helyett háromba kirajzoltatni a térképet, mivel a tömbben az ehhez szükséges adatok már megvannak az algoritmusnak hála.

Ezentúl lehet összefüggő mezőkből különleges helyeket alkotni, vegyük például egy 5 x 5-ös alföld mezőkből álló négyzet helyére egy települést generálni. Ehhez valószínűleg legalább mégegyszer végig kellene nézni a tömböt, de meg lehetne oldani effektíven.

Egy külön ablakban vagy akár a konzolban generálás előtt az adatok lekérése, hogy mekkora térképet szeretne a felhasználó, valamint, hogy mekkora arányban forduljon elő víz és szárazföld. Ezt továbbgondolva azt is meg lehetne kérdezni, hogy inkább sivatagos, hegyvidéki vagy óceán parti térképet kér.

A térkép ablakának nagyítását, kicsinyítését és teljesképernyőbe tevése is egy lehetőség, ami hasznos lehet a felhasználó számára. Fontos, hogy a mezők is alkalmazkodjanak az ablak méretének változásához.

Referenciák

- Térképészetről:
    - https://hu.wikipedia.org/wiki/T%C3%A9rk%C3%A9p%C3%A9szet
- C++ nyelvről:
    - https://hu.wikipedia.org/wiki/C%2B%2B#A_C++_alapelemei
- SDL-ről:
    - https://hu.wikipedia.org/wiki/Simple_DirectMedia_Layer
    - https://lazyfoo.net/tutorials/SDL/index.php
- Diamond-square algoritmusról:
    - https://en.wikipedia.org/wiki/Diamond-square_algorithm