Zajos képek dokumentáció

*Készítette: Gergácz Dániel*

*Neptun: FAZXPD*

*E-mail:* [*gergaczdaniel@gmail.com*](mailto:gergaczdaniel@gmail.com)

*Dátum: 2014. január 06.*

# Feladat

Adott egy fekete-fehér információ (szöveg, szimbólum, kézírás), melyet kép formátumban tárolunk. Készítsünk olyan programot, mely a bemeneti képből két "zaj" fóliát készít. A célunk az, mindkét fólián olyan fekete és fehér (átlátszó) négyzetek szerepeljenek, melyek szabad szemmel véletlennek tűnnek (az információ nem nyerhető ki egyetlen fóliából), de a két fóliát egymásra helyezve az információ feketén kirajzolódik. Az elkészített programban legyen lehetőség a következő funkciókra:

* A zaj fóliákon szereplő négyzetek mérete legyen választható.
* Legyen lehetőség két fóliából visszafejteni az eredeti információt (a fölösleges zajt eltűntetve).
* Egy több képből álló információt tudjunk úgy titkosítani, hogy minden képhez (kettő helyett) csak egy zaj fóliát készítünk, és a titkosítás feloldásához egyetlen mesterfóliát készítünk, mely bármelyik képen található információt felfedi.

Az előadáson bemutatott két minta fólia letölthető a http://people.inf.elte.hu/fekete/alg\_msc/zajos\_kepek/ címről.

# Megvalósítás

A feladat megvalósítása HTML5 és JavaScript segítségével történt, főként az bemeneti adatok széleskörű megadása miatt, ugyanis a következő opciók közül választhatunk:

* kép rajzolása
* már meglévő kép betöltése
* szöveg megadása, amit képpé tudunk konvertálni

A megvalóstás ehhez mérten három részre osztható, az input generálása, zajos kép generálása, és zajos kép dekódolása. A generált zajos képet, az input és az output megjelenítése egy-egy HTML5-ös canvas-on történik, ugyanis ez lehetővé teszi a rajta elhelyezett információ pixelenként történő feldolgozását, valamint lehetőség van képként való mentésre is a későbbi betöltés megvalósítása végett.

A zajos fóliákat kétféleképpen állíthatjuk elő az ’Encode image’, valamint az ’Encode with existing key’ feliratú gombokkal. Utóbbi használata esetén betölthetünk egy már meglévő kulcsot, vagy ellenkező esetben az előzőleg használt kulcsot fogja használni a program. A zajos képek előállítása előtt még lehetőségünk van megadni a generált képeken lévő pixelek nagyságát. Amennyiben ez eltér egytől, akkor számolnunk kell a dekódolt képen lévő információvesztéssel, ugyanis ez a minőség romlását eredményezi.

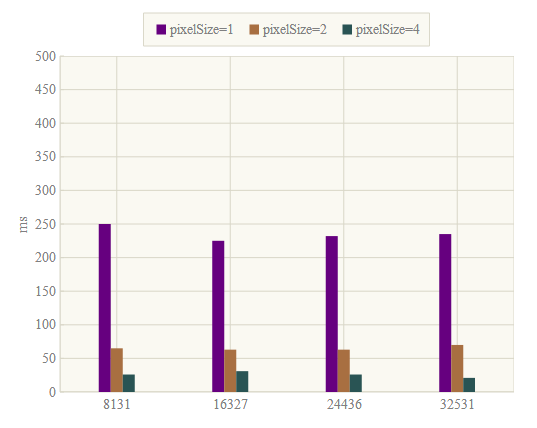
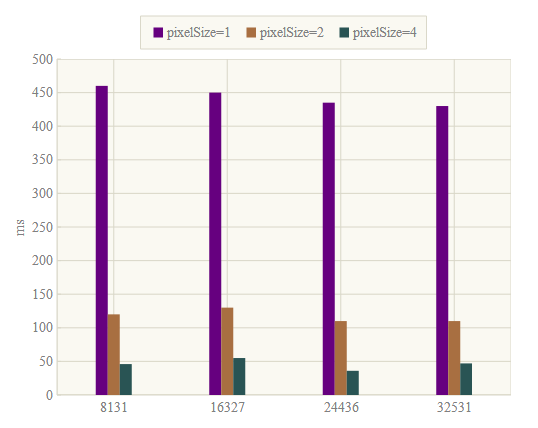
Zajos képek előállítását az ImageEncoding objektummal tudjuk végrehajtani. Először meg kell adnunk a felhasználni kívánt „canvas”-okat a set metódussal (bemeneti, zajos kép, kulcs). Ezt követően két opció közül választhatunk (teljes generálás, vagy meglévő kulccsal). A két algoritmus lényegi része megegyezik. Végig megyünk egyesével a pixeleken, 1-nél nagyobb pixel méret esetén az adott pixelméretbe eső pixeleken. Itt eltároljuk, hogy az adott négyzeten hány darab minősíthető feketének és fehérnek (átlátszó). Az algoritmus egyszerűsítése végett, akkor minősítünk feketének egy pixelt, ha fekete a színe, és nem 0 az átlátszósága, ellenkező esetben fehérnek minősítjük. Egy vizsgált négyzetet akkor minősítünk feketének, ha több fekete pixel helyezkedik el rajta, mint fehér.  
Ezt követően annyi a dolgunk, hogy meghatározzuk, hogy a zajos képen, valamint a kulcson milyen mezőnek felel meg az így kiszámolt négyzet.  
Ahhoz hogy ne lehessen kinyerni egyik zajos képből se az eredeti információt, véletlenített algoritmust használunk. Fekete esetén kiválasztjuk, hogy melyik kimeneti képre kerüljön a fekete, a másikon az adott hely üresen marad. Fehér esetén kiválasztjuk, hogy fehér vagy fekete legyen mindkettő. Az így előállított képek garantálják, hogy ne lehessen kinyerni bármelyik képből az információt.

Már meglévő kulcs esetén teljesen analóg módon hajtjuk végre az algoritmust, ami annyit tesz, hogy az utolsó leírt lépésnél figyelembe vesszük, hogy a kulcson milyen színű a négyzet, és ez alapján színezzük a zajos képet.

Létező kulcs esetén figyelni kell arra, hogy a kulcson azonos legyen egy pixel mérete, mint az éppen beállított, mivel erre külön nem figyel a program, ugyanis külön egy kulcsról nem lehet megmondani, hogy mekkora pixelmérettel lett generálva. Eltérő pixelméret esetén könnyen generálható olyan kép, amelyből részlegesen kinyerhető az eredeti információ. Valami ebben az esetben a dekódolt kép sem garantált, hogy zajmentes lesz, és az eredeti információt fogja ábrázolni. Ugyanis a dekódolás egyesével lépked végig a pixeleken, így neki nincs szüksége arra az információra, hogy hány pixellel voltak generálva a képek, egyszerűen csak annyit csinál, hogy két fehér, vagy két fekete pixel esetén fehér pixelt eredményez, eltérő szín esetén pedig feketét.

# Hatékonyság vizsgálata

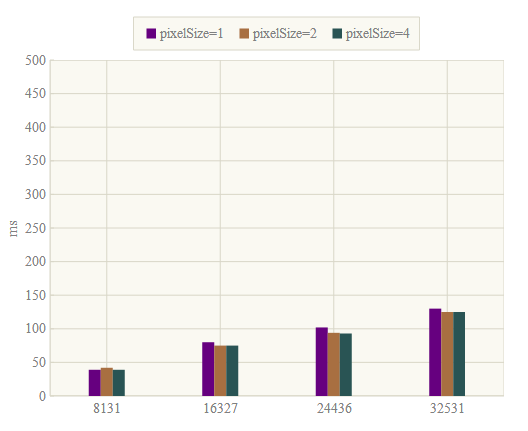
Az algoritmusok hatékonyságának vizsgálatánál a legköltségesebb művelet a fekete pixelek kirajzolása, ugyanis a fehér (átlátszó) pixelek esetén nincs semmi dolgunk, mivel ezek alapértelmezetten átlátszóak. Az algoritmus futási idején javíthatunk, ha nagyobb pixelméretet választunk, mivel ha például 4-es nagyságú pixel méretet választunk, akkor egy fekete pixelt egyszerre rajzolunk ki, tehát 16 valódi pixelt egy művelettel színezünk be. A kódolási költségen javíthatunk még úgy, hogy csak a zajos képet generáljuk le egy megadott kulcs alapján. Ezeket kiolvashatjuk az 1. ábrából és a 2. ábrából. A teszteket egy 400px x 300px méretű canvas-on végeztük, a JavaScript beépített dátum objektumának segítségével. Az ábrák a függőleges tengelyen mutatják az algoritmus időigényét, a vízszintes tengelyről pedig a bemeneti képen található fekete pixelek számát olvashatjuk le. Mind a 4 teszt esetet lefuttattuk 1-es, 2-es, és 4-es zajos képen lévő pixelmérettel.



0.1. ábra: Bemeneti kép elkódolása (meglévő kulccsal) 0.2. ábra: Bemeneti kép elkódolása (2 képet létrehozva)

Az ábrák alapján jól látható, hogy a bemeneti képen szereplő fekete pixelek számától nem függ a futási idő, ez egyszerűen azért van, mert ettől függetlenül még létre kell hoznunk a zajokat a kimeneti képeken.

A 3. ábrán a dekódolás futási idejét mértük le. Itt már jól látható, hogy függ a fekete pixelek számától, viszont nem függ a zajos képeken lévő pixelmérettől, ez pontosan a korábban leírtak miatt van, azaz ez az algoritmus nem figyeli külön a pixelméretet.



3. ábra: Kép dekódolása

# Következmény

A megfigyelések alapján azt állapíthatjuk meg, hogy megéri növelni a pixelméretet, ha nem annyira kidolgozott képet szeretnénk elkódolni, azaz visszafejtés után kiolvasható marad az információ, ezzel ugyanis sok számítási időt takaríthatunk meg. A mi példánkban ez a 10-edére csökkent, amikor 1-es pixelméret helyett 4-es pixelméretet alkalmaztunk.  
A dekódolás költsége viszont erősen függ a kép információ tartalmától.