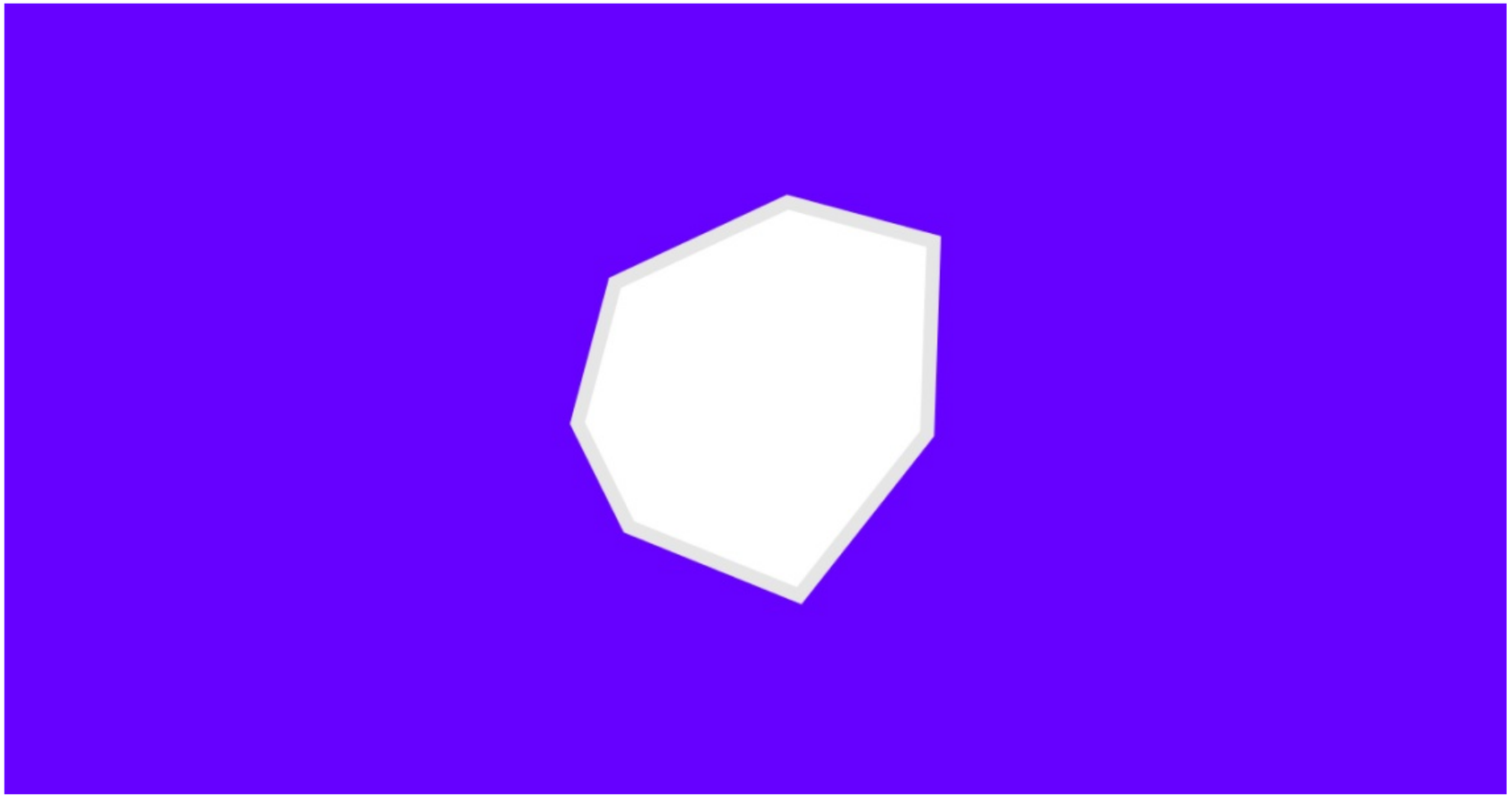


Generator Asteroido-podobnych Obiektów 2D (Generator Wielokątów) (sheadovas/algorytmy/generator-asteroido-podobnych-obiektow-2d-generator-wielokatow/)

Sty 02, 2016 / Algorytmy (sheadovas/category/algorytmy/)

Prosty algorytm służący do generowania pseudo-losowych asteroid (wielokątów).

Opis

Algorytm służy do wygenerowania wielokąta, mogącego imitować asteroidę w grze 2D.

Idea algorytmu polega wygenerowaniu *N* współrzędnych wierzchołków, których odległość od punktu generowania wszystkich punktów *S* (czyli w rezultacie środka koła) jest nie większa od jego promienia *R* (w przypadku wielokąta wypukłego odległość wygenerowanego punktu od *S* jest zawsze równa *R)*.

Algorytm

Wejście

- N – ilość wierzchołków, z których będzie składał się wielokąt ($0 < N \leq 360 \in \mathbf{N}$);
- S – punkt względem, którego generowana jest reszta wierzchołków;
- R – maksymalna odległość punktów od S ($R > 0 \in \mathbf{N}$);
- *isConvex* – flaga *true*/*false* informująca o tym czy wielokąt ma być wypukły.

Wyjście

- $x[N]$ – kolekcja (tablica/lista) punktów będącymi wierzchołkami wielokąta (są ułożone zgodnie z ruchem wskazówek zegara).

Zmienne/funkcje pomocnicze

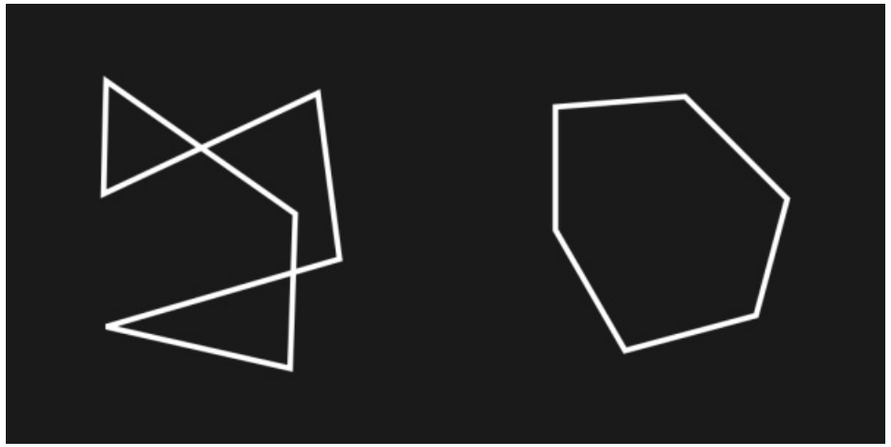
- i – aktualnie generowany wierzchołek;
- d – odległość (od S) w jakiej ma zostać wygenerowany wierzchołek;
- φ – kąt wycinka koła;
- ψ – kąt pod jakim zostanie wygenerowany nowy wierzchołek;
- *Random(min, max)* – funkcja generująca liczbę całkowitą z przedziału [min,max).

Lista kroków

1. $i = 0$, $\varphi = 360 / N$, $d = R$.
2. Jeżeli *isConvex* jest równy:
    - *false* to $d = Random(0,R)$.
3. $\psi = Random(0, \varphi) + i * \varphi$.
4. $x[i] = S + (cos(\psi) * d, sin(\psi) * d)$.
5. $i = i + 1$.
6. Jeżeli $i < N$, to wróć do punktu *(2)*.

Wyjaśnienie działania

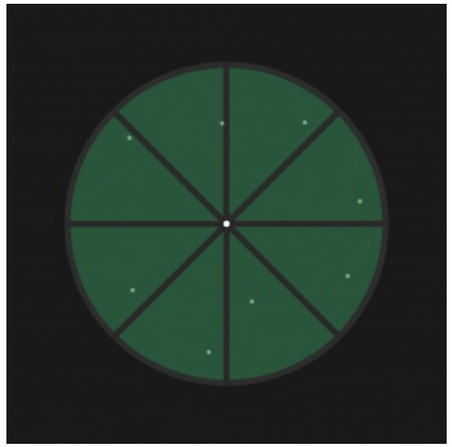
Naszym celem jest wygenerowanie ‚ładnego’ wielokąta, tzn takiego którego krawędzie nie przecinają się ze sobą. Po lewej stronie widzimy efekt, którego nie chcemy, a po prawej efekt którego pożądamy.



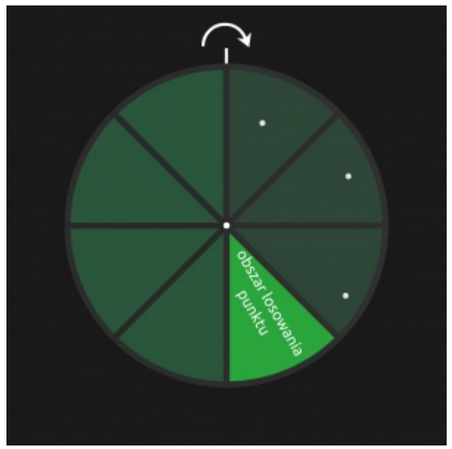
Pierwszą intuicją jaką należy zauważyć jest fakt że tak naprawdę generujemy obiekt znajdujący się wewnątrz koła (środkiem koła jest punkt względem, którego tworzymy wielokąt, a promieniem jest maksymalna odległość wierzchołków od punktu startowego), a jedynie co robimy to wybieramy w sposób losowy punkty, które do niego należą.

Kolejnym ważnym faktem jest to, że musimy ustalić sobie jakiś kierunek względem którego będziemy ustawiali kolejne wierzchołki (kierunek zgodny lub przeciwny do ruchu wskazówek zegara), tylko w ten sposób połączenia punktów nie będą się przecinały.

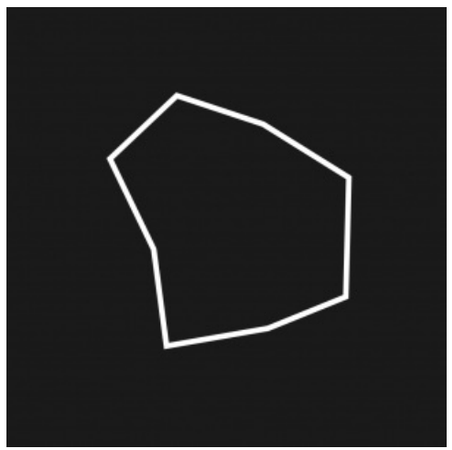
Skoro pracujemy na kole, to sprawa staje się prosta. Pierwszym krokiem algorytmu jest podzielenie okręgu na N równych części, w rezultacie nasze koło składa się z N wycinków (kawałków pizzy).



Każdy „kawałek pizzy” to zbiór punktów (inaczej mówiąc: obszar z którego możemy wybrać punkt), z których możemy wybrać losowy punkt (krok 2 i 3) i dodać go do kolekcji (krok 4).



Dalej analogicznie przechodzimy po wszystkich wycinkach (wg wcześniej ustalonej kolejności) i dla każdego wycinka wybieramy losowo jeden punkt. Na koniec łączymy wszystkie punkty i cieszymy się z otrzymanego rezultatu.



Implementacja

C/C++

Implementacja w języku C++ C++

```cpp
/*************************************************************

    2D Asteroids Generator (Vertices Generator)

    License:
        BSD-3 ( sheadovas/bsd-3-license/ )

    Author:
        Szymon Siarkiewicz
        sheadovas/
        kontakt@szymonsiarkiewicz.pl

*************************************************************/


#include <cmath>
#include <cstdlib>

#define M_PI 3.14159265358979323846

using namespace std;

/*
    Klasa pomocnicza - przechowuje pozycje punktow
*/
class Vector2
{
public:
 Vector2()
 {
 x = y = 0;
 }

```

Demo

Uwaga, Demo nie było przetestowane!

Demo | C++ & SFML C++

```
/*************************************************************

Demo 2D Asteroids Generator (Vertices Generator)

License:
BSD-3 ( sheadovas/bsd-3-license/ )

Author:
Szymon Siarkiewicz
sheadovas/
kontakt@szymonsiarkiewicz.pl

*************************************************************/



#include "AsteroidGenerator.h"
#include <SFML\Graphics.hpp>

#define WIDTH 640
#define HEIGHT 640

#define POINT_COUNT 15
#define RADIUS 25



void CreateAsteroid(sf::ConvexShape& shape, bool isConvex)
{
 shape.setPointCount(0);
 shape.setPointCount(POINT_COUNT);
 shape.setFillColor(sf::Color(Random(255), Random(255), Random(255)));
 shape.setPosition(0, 0);
```

Code ON!