# Zadanie 1 – LGA


## Kacper Hyla

Grupa 1


Zadanie polegało na przeprowadzeniu symulacji gazu za pomocą automatu komórkowego. Do zaprogramowania zadania użyto języka C++ oraz biblioteki SFML celem wizualizacji wyników. Jako IDE użyto programu Microsoft Visual Studio w wersji 2022.

## Program:

Automat składa się z komórek, których definicja wygląda tak, jak pokazano poniżej.

```
class cell {
public:
	int in;
	int out;
	int is_a_wall;

	cell() {
		in = 0;
		out = 0;
		is_a_wall = 0;
	}

	void solidify() {
		is_a_wall = 1;
	}

	void remove() {
		is_a_wall = 0;
	}
};
```

Zmienna in oraz out odpowiada za mechanizm streamingu, natomiast is_a_wall decyduje, czy dana komórka jest przeszkodą.

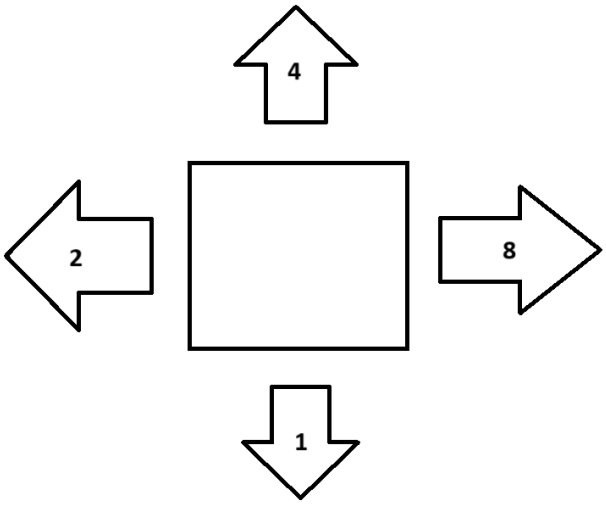
Klasa automatu skończonego zaprezentowana jest poniżej.

```
class CelluarAutomata {
	int width, height;	//both variables describe amount of cells, not size
in pixels!
	cell** cells;

public:
	CelluarAutomata(int, int);
	void display();
	void fill(int units, int up, int down, int left, int right);
	void setup();
	void execute_iteration();
};
```

Poza konstruktorem, który zwyczajnie rezerwuje pamięć na przechowanie wszystkich komórek, ważne są funkcję setup oraz fill, które kolejno, wyznaczają ściany w automacie, oraz losują pozycję gazu w podanym zakresie.

```
void CelluarAutomata::setup() {
	for (int i = 0; i < width; i++) {
		cells[0][i].solidify();
		cells[height - 1][i].solidify();
	}

	for (int i = 0; i < height; i++) {
		cells[i][0].solidify();
		cells[i][width-1].solidify();
		cells[i][width / 4].solidify();
	}
}


void CelluarAutomata::fill(int units, int up, int down, int left, int right) {

	srand(time(NULL));
	for (int i = 0; i < units; i++) {
		int x = up + rand() % (up - down);
		int y =  left + rand() % (right - left);
		if (cells[x][y].out > 0) {
			i--;
			continue;
		}
		int vel = rand() % 4;
		if(vel == 0)cells[x][y].out += 1;
		if (vel == 1)cells[x][y].out += 2;
		if (vel == 2)cells[x][y].out += 4;
		if (vel == 3)cells[x][y].out += 8;

	}
}
```

Warto tutaj powiedzieć coś o zmiennych in oraz out każdej komórki. Przyjmują one wartości z przedziału 0 – 15, czyli wartości możliwe do zapisania na 4 bitach. To właśnie pozwala jednoznacznie określić, skąd (i w którą stronę) poruszają się komórki.



Przykładowo, wartość in równa 10 oznacza, że gaz wlatuje do komórki z lewej i prawej strony.

Sama operacja streamingu następuje w sposób opisany poniżej.

Każda komórka aktualizuje stan zmiennej in swoich sąsiadów w oparciu o swoją wartość zmiennej out. Następnie na podstawie informacji, czy dana komórka jest ścianą, lub czy zachodzi kolizja, ustalanie jest parametr out, który zostanie wykorzystany w kolejnej iteracji. Wszystko to zawarto w funkcji execute_iteration.

```
void CelluarAutomata::execute_iteration() {
	for (int i = 0; i < height; i++) {
		for (int j = 0; j < width; j++) {
			int mod = cells[i][j].out % 2;

			if (mod)cells[i+1][j].in += 1;
			cells[i][j].out /=  2;
			mod = cells[i][j].out % 2;
			if (mod)cells[i][j - 1].in += 2;
			cells[i][j].out /= 2;
			mod = cells[i][j].out % 2;
			if (mod)cells[i-1][j].in += 4;
			cells[i][j].out /= 2;
			mod = cells[i][j].out % 2;
			if (mod)cells[i][j + 1].in += 8;

		}
	}

	for (int i = 0; i < height; i++) {
		for (int j = 0; j < width; j++) {
			if (cells[i][j].is_a_wall == 1) {
				if(cells[i][j].in == 8)cells[i][j].out = 2;
				if(cells[i][j].in == 2)cells[i][j].out = 8;
				if (cells[i][j].in == 4)cells[i][j].out = 1;
				if (cells[i][j].in == 1)cells[i][j].out = 4;
				cells[i][j].in = 0;
			}
			else {
				if (cells[i][j].in == 10)cells[i][j].in = 5;
				else if (cells[i][j].in == 5)cells[i][j].in = 10;
				cells[i][j].out = cells[i][j].in;
				cells[i][j].in = 0;
			}

		}
	}
}
```

W pierwszej pętli następuje zmiana wejścia sąsiadów. Dzięki operacją modulo 2 oraz dzielenia przez 2 bez reszty można bez problemów ocenić, które komórki należy aktualizować (pozwala ocenić czy na danej pozycji w czterobitowej liczbie znajduje się 0, czy 1).

W drugiej fazie, jeżeli dana komórka jest ścianą, następuje odbicie ruchu gazu w drugą stronę, a jeżeli zaszła kolizja, wyznaczane są nowe kierunki ruchu gazów (jeżeli dwie cząsteczki weszły do komórki z lewej i prawej, to wyjdą one z góry i z dołu i vice versa).

Wizualizacja działania programu odbywa się w funkcji display. Tworzy ona okno SFML i tworzy teksturę, która odpowiada stanowi automatu, gdzie czerwone komórki to gaz, a szare – ściany. Okno ma rozmiary automatu pomnożone przez 3, zatem jednej komórce odpowiada 9 pixeli. Zastosowano zatem dodatkowe małe pętle wewnątrz głównych pętli(gdzie główna oznacza iterującą się przez cały automat), które odpowiadają za pokolorowane wszystkich (lub części) pikseli reprezentujących komórkę. Następnie na podstawie tekstury jest tworzony sprite, który jest wyświetlany w oknie.

```
void CelluarAutomata::display() {
	int h_pixels = height * 3;
	int w_pixels = width * 3;
	sf::RenderWindow window(sf::VideoMode(w_pixels, h_pixels), "LGA");

	sf::Image canvas;
	sf::Texture texture;
	sf::Sprite sprite;

	canvas.create(w_pixels, h_pixels);

	for (int i = 0; i < height; i++) {
		for (int j = 0; j < width; j++) {
			if (cells[i][j].out > 0) {
				int scaled_i = i * 3;
				int scaled_j = j * 3;
				for (int k = 1; k < 2; k++) {
					for (int l = 1; l < 2; l++) {
						canvas.setPixel(scaled_j + k, scaled_i + l,
sf::Color(255, 50, 50));
					}
				}
			}
			if(cells[i][j].is_a_wall == 1) {
				int scaled_i = i * 3;
				int scaled_j = j * 3;
				for (int k = -1; k < 2; k++) {
					for (int l = -1; l < 2; l++) {
						if (scaled_i + k < 0 || scaled_i + k >=
h_pixels || scaled_i + l < 0 || scaled_i + l >= w_pixels)continue;
						canvas.setPixel(scaled_j + k, scaled_i + l,
sf::Color(100, 100, 100));
					}
				}
			}

		}
	}
	texture.loadFromImage(canvas);
	sprite.setTexture(texture);
	window.clear();
	window.draw(sprite);
	window.display();
```

Pozostaje główna pętla programu, która po uruchomieniu czeka na wciśnięcie jednego z dwóch przycisków. Spacji – która wykonuje iteracje, oraz aktualizuje dane na ekranie, oraz klawisza ‘A’, który zdejmuje część bariery, pozwalając na rozprzestrzenienie się gazu.

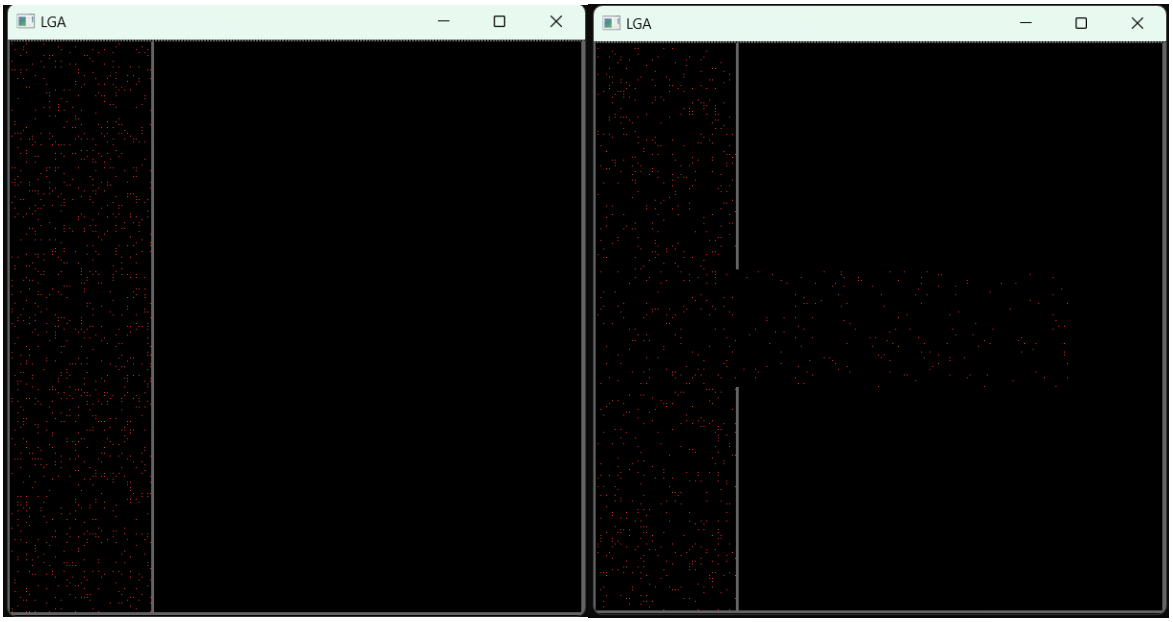
```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Space)) {
	execute_iteration();
	//Te same mechanizmy wyświetlania danych na ekranie, co powyżej


if (sf::Keyboard::isKeyPressed(sf::Keyboard::A))
	for (int i = 80; i <= 120; i++) {
		cells[i][width / 4].remove();
	}
```

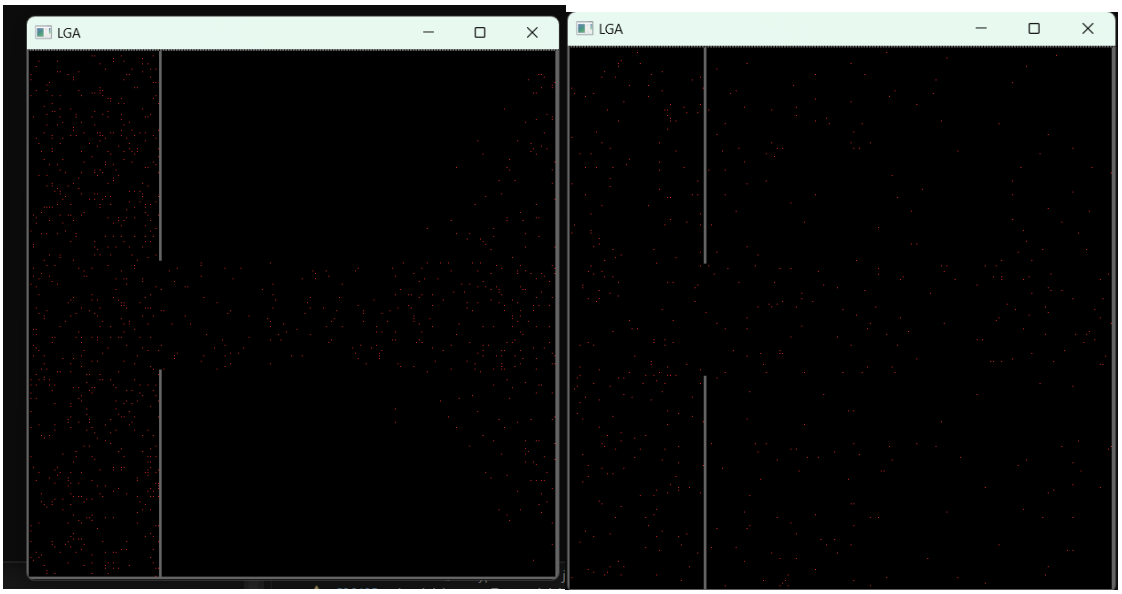
Pozostaje funkcja main, która tworzy obiekt automatu, inicjalizuje go i wyświetla wynik.

```
int main()
{

    CelluarAutomata a(200, 200);
    a.setup();
    a.fill(1500, 1,199,1,49);
    a.display();


    return 0;
}
```

Tutaj wartościami początkowymi było 1500 jednostek gazu, generowanych wewnątrz ¼ automatu (pierwszej z 4 kolumn).

Oto zrzuty ekranu, przedstawiające działanie programu.




LGA
LGA