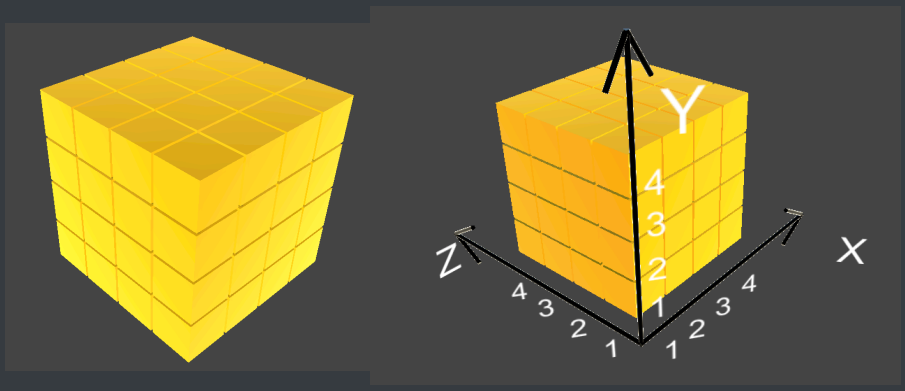
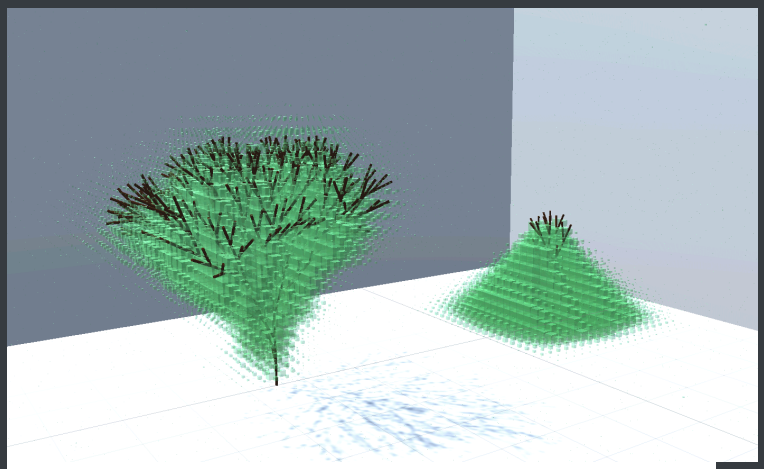
Voxel space

https://www.cc.gatech.edu/~turk/bio_sim/articles/voxel_space_automata_89.pdf

Voxel space to kwantyzowana przestrzeń 3D podzielona na voxele czyli sześciany identycznej wielkości.



Do każdego sześcianu można przypisać wartość liczbową i w ten sposób przechowywać jakąś informację o tym regionie (zawartość substancji chemicznej, wilgotność, zacienienie, obecność innych obiektów). Można to wykorzystać do rozmieszczania obiektów w przestrzeni, unikania kolizji, ustalania zacienienia w przestrzeni.



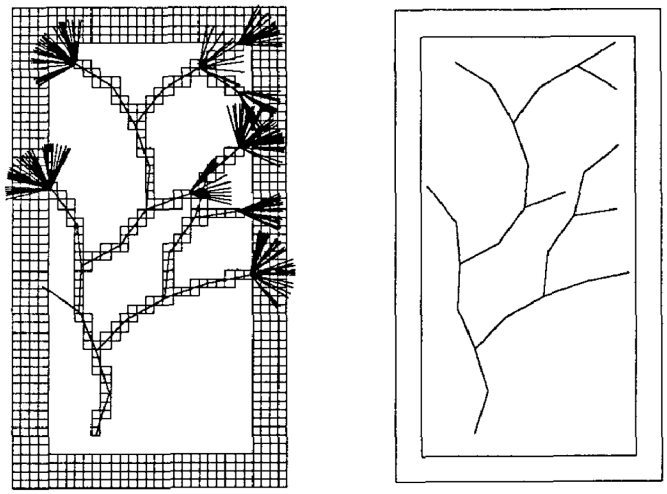


Figure 2.

Skeleton generated by tree-structured random walk through 2D voxel space with intersection avoidance. Frame at left shows all attempts to place segments and voxels intersected by successfully placed segments. Fan-shaped clusters represent unsuccessful trials. Skeleton generated is shown at right.

Figure 5.

This image was created by estimating diffuse reflection at occupied voxels and then dicing the polygonal model of the scene to the voxel grid, assigning each fragment the color of the corresponding voxel. There are approximately 27,000 polygons in the scene and image generation took roughly 30 hours on a sun4. The color table is nonlinear, making dark areas appear brighter.

Voxel space wymaga większej ilości RAMu (trzeba te wszystkie wartości przechowywać), ale jako, że nie trzeba wszystkiego na żywo wyliczać, tylko wczytuje się dane z tablicy, to jest szybka metoda.

Przykład – mrówki

Feromony

Fermony to substancje, za pomocą których organizmy przekazują sobie informacje.

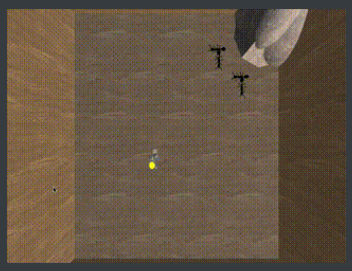
Opis problemu

Przyjmujemy, że w prawdziwym świecie, mrówki poruszają się w sposób losowy; gdy znajdują pożywienie, wracają do swojej kolonii pozostawiając ślad składający się z feromonów. Gdy inna mrówka natknie się na ten ślad, przestaje poruszać się w sposób losowy i podąża za śladem w kierunku pożywienia.

Jednak po pewnym czasie feromony wyparowują, a więc siła ich działania maleje. Im dłuższa jest trasa od pożywienia do kolonii, tym więcej mają czasu feromony, aby wyparować. Krótsze trasy jednak zapewniają, iż siła działania feromonów będzie większa. Parowanie feromonów jest efektem pozytywnym, bowiem pozwala to na odnajdywanie optymalnej trasy do pożywienia. Gdyby feromony nie wyparowywały, każda kolejna trasa miałaby taką samą siłę jak poprzednia, przez co nie dochodziłoby do odnalezienia optymalnego rozwiązania problemu.

Zatem gdy jedna mrówka odnajdzie dobrą (krótką) drogę, inne mrówki będą podążać tą właśnie drogą, również zostawiając feromony, a więc zwiększając ich natężenie. Ostatecznie wszystkie mrówki będą poruszać się tą samą, najlepszą drogą, a pozostałe drogi zostaną zapomniane (wyparują). (Wikipedia)

Przykład rozwiązania



https://projects.duszekjk.com/ants1.0/

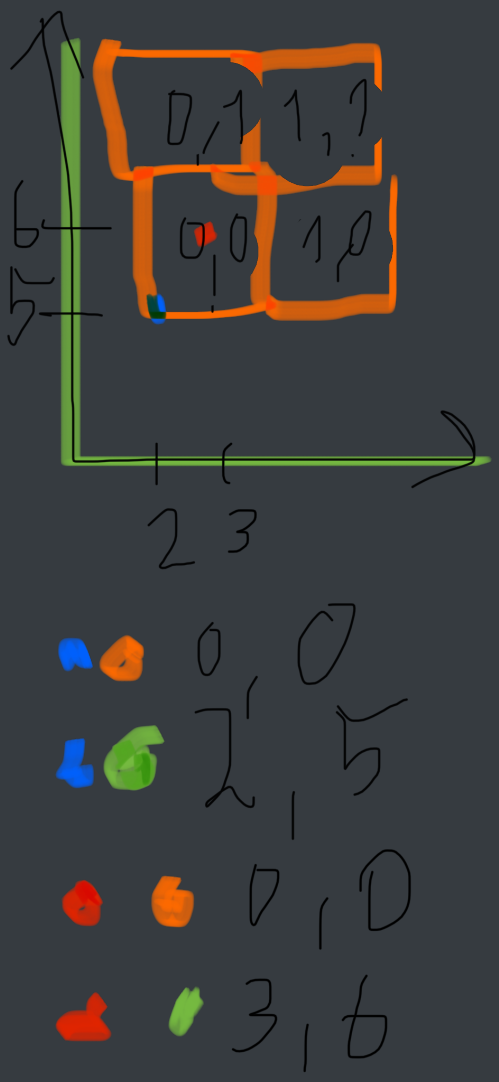
Zadanie 1

1. Otwórz projekt voxelSpace w Unity
2. Otwórz scenę "Scenes/theAntsAreMarching"
3. Zapoznaj się z przygotowanymi skryptami
    1. Ant (obiekt mrówki)
        1. **bool** hasFood - czy ma jedzenie
        2. **sbyte** foodSearch - stan szukania jedzenia, 1 wraca do mrowiska, -1 idzie od mrowiska, szuka jedzenia
        3. **VoxelSpace** voxelSpace
        4. **void** sniff() - ustala kierunek poruszania się
        5. **void** turnVerticaly() - zmienia kierunek poruszania się mrówki na wertykalny (góra - dół)
        6. **void** turnHorizontaly() - zmienia kierunek poruszania się mrówki na horyzontalny (lewo - prawo)
        7. **void** go() - mrówka idzię o odległość odpowiednią dla jednej klatki animacji
    2. VoxelSpace (reprezentacja voxel space, którą będziemy robić)
        1. **void** Update() - co odpowieni czas wywołuje funkcje sniff, go na mrówkach, oraz weakenPheromones i placeFood
        2. **void** addAnts() tworzy mrówki
        3. **void** updateLights() uaktulania wizualizacje voxel space
        4. **void** placeFood() umieszcza jedzenie w losowej pozycji
        5. Funkcje do napisania w zadaniach, opisane poniżej

Zadanie 2

1. Napisz funkcję createVoxelSpace()
    1. Zainicjalizuj atrybut `byte[,] voxels`, to znaczy utwórz tablicę dwuwymiarową o wymiarach `sizeX` na `sizeY`.
    2. W pętli ustaw wartość każdej komórki na 0.
    3. Do wizualizacji tej przestrzeni wykorzystamy żółte kule, których rozmiar będzie zależny od zawartości feromonu. Utwórz dwuwymiarową tablicę typu `GameObject` o takich samych wymiarach jak powyżej i przypisz ją do atrybutu `GameObject[,] lights`.
    4. W pętli zapełnij tablice instancjami `previewShape`, do utworzenia nowej instancji obiektu służy funkcja `Instantiate(GameObject gameObject)`. Ustaw każdemu pozycję odpowiadającą pozycji danego voxela korzystając z funkcji `positionInWorld` (uzupełnimy ją w następnym kroku).

2. Napisz funkcje **positionInVoxelSpace**(Vector3 worldPosition) i **positionInWorld**(Vector2 voxelPosition) skalujące współrzędne z ciągłych (współrzędnych gameObject'ów - floats), na te w voxel space (wartości współrzędnych są wartościami Int o wielkości od 0 do odpowiednio sizeX-1 i sizeY-1) i z powrotem na ciągłe. Weź po uwagę różne skale (voxelScale), oraz umiejscowienie obiektu table (gameObject.transform.position).
   Jako, że w pozycji ciągłej (obiektów gry) mamy 3 wymiary, a ten voxel space ma 2, to ustaw wymiary xy z voxel space w wartościach xz, a wymiar y pomiń/ustaw na 0.



1. Po dodaniu tej funkcji, w prawym górnym rogu pojawi się mrowisko
2. Dodaj do funkcji z poprzedniego zadania pozycje obiektów lights wykorzystując nowe funkcje

3. Napisz funkcje getPheromoneAt(**foat** x, **float** y)
    1. Funkcja przyjmuje współrzędne w przestrzeni świata i ma zwrócić zawartość fermonu przechowywaną w *voxel space*.
    2. Przekonweruj współrzędne x,y do przestrzeni voxeli
    3. Jeśli te współrzędne się mieszczą w voxel space, to zwróć odpowiednią wartość (uwzględniając zmiane współrzędnych na voxelowe). Jeśli nie - zwróć **-1**
4. Napisz funkcje sniff() (skrypt Ant)

Mrówki po wyjściu z mrowiska poruszają się w kierunkach w dół lub w lewo. Jeśli znajdą jedzenie, to zawracają i idą w kierunku mrowiska (prawo i w górę). Jeśli mrówka dojdzie do krańca voxel space, to też zaczyna iść w kierunku mrowiska. Mrówka po dojściu do mrowiska zostawia jedzenie (jeśli je ma) i znowu wychodzi z mrowiska. Kierunek (dół i lewo lub góra i prawo) jest zależny od zawartości feromonu w sąsiednich voxelach w tym kierunku. Jeśli zawartość fermonu jest taka sama, to idzie w losowym kierunku.

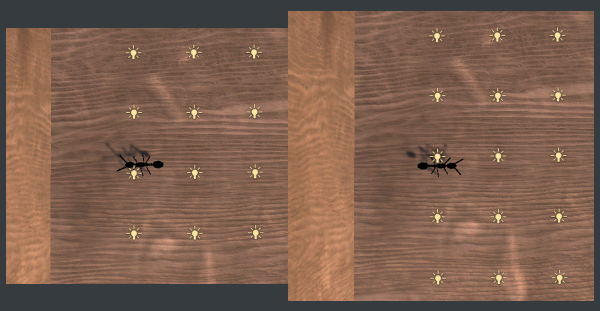
https://projects.duszekjk.com/ants1.0/

bool hasFood - czy ma jedzenie

sbyte foodSearch - stan szukania jedzenia, **1** wraca do mrowiska, **-1** idzie od mrowiska, szuka jedzenia

1. Funkcja ta ma na celu zmianę kierunku poruszania się tej mrówki
2. Sprawdź i uaktualnij status (ten voxel)
    1. Jeśli jeśli funkcja **getPheromoneAt** zwraca liczbę większą od **250**, to to jest jedzenie, z którym mrówka ma wracać do mrowiska (ustawiamy **hasFood** i **foodSearch**)

    2. Jeśli jesteśmy w punkcie **(0,0)** (voxel space) to jesteśmy w mrowisku (zostawiamy jedzenie i zmieniamy kierunek na szukanie jedzenia (ustawiamy **hasFood** i **foodSearch**))
    3. Jeśli dojdziemy do końca voxelSpace, to wracamy do mrowiska (ustawiamy tylko foodSearch)



3. Ustal kierunek (sąsiednie voxele)

    1. Sprawdź czy zawartość fermonu jest większa jeśli się pójdzie horyzontalnie (prawo, lewo), czy wertykalnie (góra, dół) i skieruj mrówkę w odpowiednim kierunku (możesz wykorzystać **foodSearch**, żeby wiedzieć gdzie sprawdzać)
    
    2. jeśli są takie same, to idź w losowym kierunku
    3. Podpowiedź - funkcje: **turnVerticaly()** i **turnHorizontaly()**
4. Jeśli mrówka ma jedzenie, to zwiększ w obecnej pozycji wartość voxel'a o **40**. (ten voxel)

5. Napisz funkcje updateVoxelSpace(), która obniża wartości feromonów jako wynik parowania w czasie
    1. Przejdź po wszystkich voxelach, które mają wartość poniżej **250** (nie są jedzeniem) i obniż ich wartość o **1**
    2. Minimalna możliwa wartość to **0**
6. Sprawdź jak mrówki się poruszają
    1. Jeśli zadania były poprawnie zrobione, to powinny najpierw iść losowo, jak nie ma feromonów, a potem jak są, to podążać za feromonami

Zadanie 3

1. Otwórz scenę menu
2. Podpisz się w Canvas/get/Podpis
3. Wyeksportuj swój projekt dla wybranego systemu
    1. File->Build Settings
    2. Wybierz platformę
    3. Build And Run