**절차적 컨텐츠 생성을 이용한 게임 개발**

**-정올반 겨울방학 자율 프로젝트-**

**Game developing with**

**Procedural Map Generation**

**2021년 2월 2일**

**선린 인터넷 고등학교**

**소프트웨어과 & 정보보호과**

**심준, 이기연, 서윤하**

**초 록**

게임에서 궁극적으로 추구하는 바는 재미이다.

재미는 신선도와 같은 여러가지에 비례하는데, 이 프로젝트에서는 유저에게 매번

신선한 게임을 제공하여 유저로 하여끔 만족도를 높이는 것에

초점을 두었다.

이 프로젝트 보고서에는 지금까지 널리 알려진 방법들을 참고해 직접 구현한 알고리즘과 해당 알고리즘이 얼마나 효율적으로 작동하는지 소개한다.

**주요어:** 절차적 컨텐츠 생성, 이진 공간 분할, 세포 자동자

**학번:**  20318 이기연, 20411 서윤하 ,20512 심준

**목 차**

**제 1 장 서 론 …………………………………………………………………………………04**

제 1 절 개 요 …………………………………………………………………………………………………..04

제 2 절 보고서의 구성 ………………………………………………………………………………………..04

**제 2 장 배경 지식 ………………………………………………………………………………05**

제 1 절 절차적 컨텐츠 생성...…………………………………………………………………………….05

제 2 절 이진 공간 분할법…….……………………………………………………………………………..05

제 3 절 세포 자동자…………….……………………………………………………………………………..06

**제 3 장 모델 설계 ………………………………………………………………………………07**

**제 4 장 실험 결과 ………………………………………………………………………………09**

제 1 절 실험 환경 ………………………………………………………………………………………………09

제 2 절 실험 결과 ………………………………………………………………………………………………09

**제 5 장 최 적 화 ………………………………………………………………………………..13**

제 1 절 알고리즘 구현 ……………………………………………………………………………………….13

제 2 절 성 능 ………………………………………………………………………………………………….13

**참고 문헌 ………………………………………………………………………………………….14**

**제 1 장 서 론**

**제 1 절 개 요**

조사해보니 같은 PCG(절차적 컨텐츠 생성:Procedural Content Generation)여도 생성 방법에 BSP(이진 공간 분할), Cellular Automata(세포 자동자), Herringbone Wang tiles, Voronoi diagram 등등 여러가지 방법이 있다는 걸 알게되었습니다.

**제 2 절 보고서의 구성**

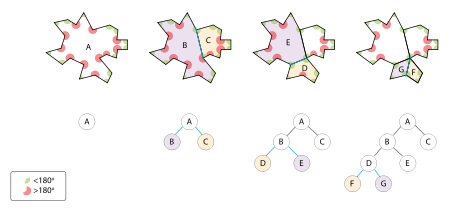
2장에서 절차적 컨텐츠 생성과 세포 자동자, 이진 공간 분할법에 대한 설명과 3장에서 모델 설계에 대해 설명하고, 해당 모델로 테스트한 결과를 4장에서 제시한다.

**제 2 장 배경 지식**

**제 1 절 절차적 컨텐츠 생성**

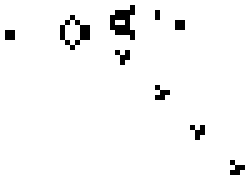
절차적 콘텐츠 생성(Procedural content generation)은 사람의 참여 없이 또는 제한된 참여만으로 콘텐츠를 생성하는 알고리즘 기법이다. 이는 최근 들어 학계와 산업계부터 큰 관심을 받고 있다. 비디오 게임을 위한 절차적 콘텐츠 생성 기법은, 무한한 반복 플레이, 새롭고 개인화된 게임 세계의 구현, 플레이어 취향의 반영, 그리고 개발 비용의 절감을 가능하게 한다. 우리는 이 중 가장 활발한 연구가 이루어지고 있는, 게임 맵 디자인에 대한 절차적 생성에 집중하고자 한다.

**제 2 절 이진 공간 분할**



A라는 가장 큰 컴포넌트가 있고 A를 B와 C로 분할하고 B를 분할하는 과정을 반복하면 된다. 위 사진에서 분할과정을 멈추는 조건은 모든각이 예각인 도형을 만드는 것이다. 이진 공간 분할 트리를 생성하면서 데이터를 관리해주면 된다. 이진 공간 분할 트리는 어느 경우에나 이진 트리 라는 것이 자명하다.   
이진 공간 분할에서 탈출 조건을 성립하는 것이 제일 중요하다. 위 사진에서의 탈출 조건은 다각형에서 모든 각이 예각인 다각형일 때 분할을 멈춘다.

**제 3 절 세포 자동자**

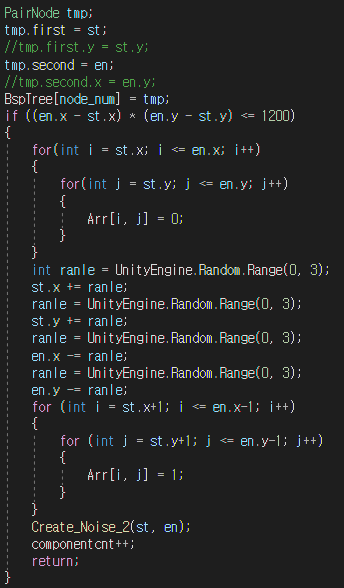
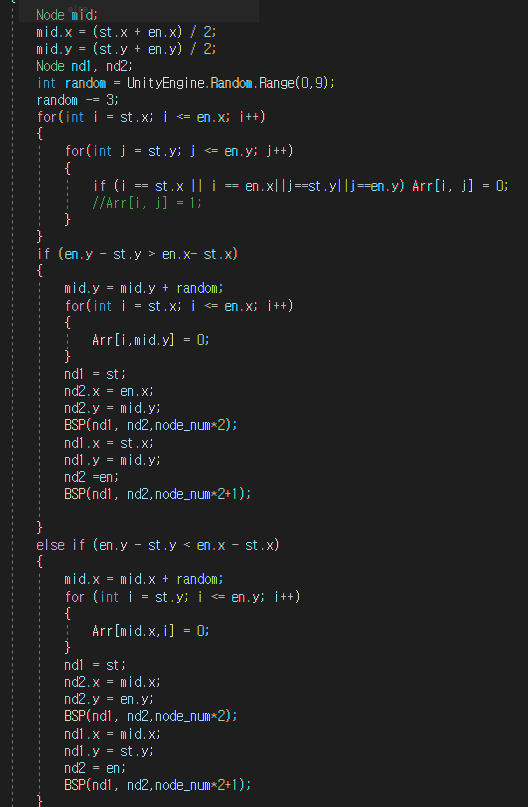


세포자동자의 정의는 '계산 가능성이론, 수학, 물리학, 복잡계, 수리생물학, 미세구조 모델링에서 다루는 이산 모형' 이라고 위키 백과에 정의되어있는데, 저렇게 설명하면 이해하기 힘들다. 하지만, 우리가 꼭 세포자동자가 계산 가능성 이론, 수학, 물리학 등등 여러분야에서 어떻게 쓰이는지는 알 필요가 아직은 없으니까, 정의에서 중심적으로 봐야할 부분 '이산 모형'이라는 부분이다.

이산 모형은 이산 수학에서 파생된 개념인데, 이산수학이란 반복되지 않은 유한,무한적인 공간에서의 수학 구조에 대해 연구하는 학문이다. 그럼 이산 모형은 반복되지 않은 유한, 무한적인 공간에서의 모델이라는걸 알 수 있다.

세포 자동자는 참/거짓 2진수로 무한/유한정의 셀(세포)들을 구성하고 있다.

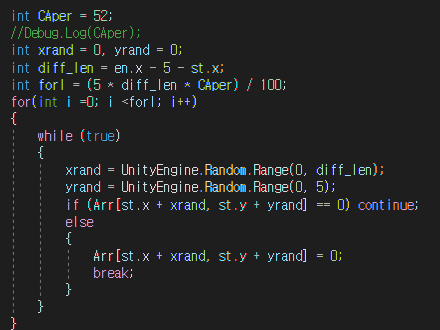
**제 3 장 모델 설계**

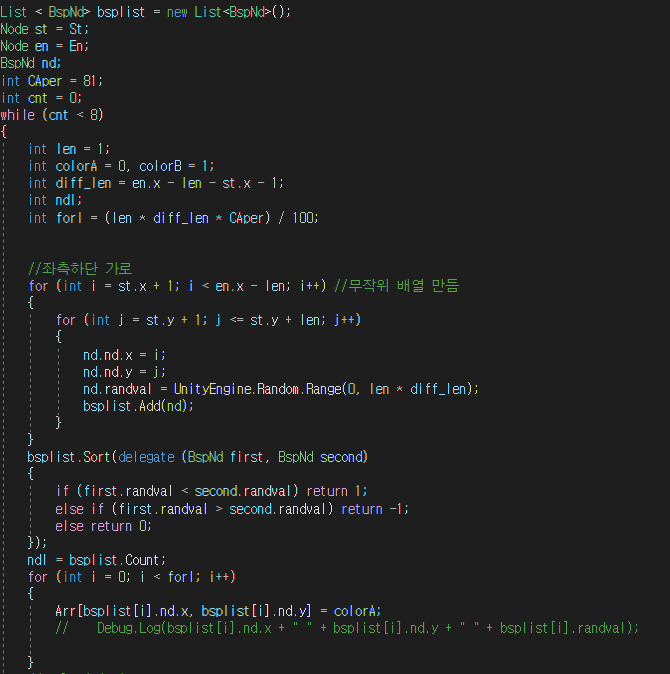
컴포넌트들의 위치와 형태를 정해주는 BSP 코드이다.. 전체 컴포넌트의 갯수를 K 라고 하자. bsptree는 기본적으로 이진트리이기 때문에 노드 갯수가 최대 N\*2-1이라는걸 알 수 있다. 또한 트리의 최대 depth가 LogN이라는 것이 자명하다.

이제 어떤 방식으로 구현 할지만 고민하면 된다. 노드 형태의 구조체를 선언하는 방법도 있고, 일차원 배열을 사용하는 방법 등등 많은 방법이 있다. 노드형태의 구조체는 구조체의 형태를 자신의 값과 자식 노드들을 저장하는 방식으로 구현해주면 된다. 이 프로젝트에서는 배열 형태를 사용하였다. 현재 노드에서 왼쪽 자식노드의 key값은 현재 노드의 key\*2, 오른쪽 자식노드의 key값은 현재노드의 key\*2+1를 저장해주면 쉽게 구현 할 수 있다.

bsptree에서 각 노드의 value 값은 컴포넌트의 시작점과 끝점으로 하였다.



세포 자동자 괴정에서 노이즈를 추가할 때 x좌표와 y좌표를 랜덤으로 생성하고, 매 쿼리마다 그 좌표가 이미 색칠 되어있는지 확인할경우 시간복잡도가 O(INF) 이상의 시간이 걸릴 수도 있기에 이 과정에서 어떻게 해야 디버깅 속도를 줄일 수 있을지 고민했다.



별 다른 알고리즘 없어 정렬을 잘 이용하면 O(NlogN)의 시간복잡도에 해결 할 수 있다. 색칠하고자 하는 구역의 좌표값들을 리스트에 넣어주면서 랜덤 값을 지정해주는데, 후에 랜덤값에 대해 오름차순으로 정렬하여 앞에서부터 차례대로 뽑아 색칠해주면 된다. 이전 방식에서 디버깅이 최소 3분 내외는 걸렸는데, 최적화를 진행준 뒤에는 모든 경우에서 20초가 걸렸다

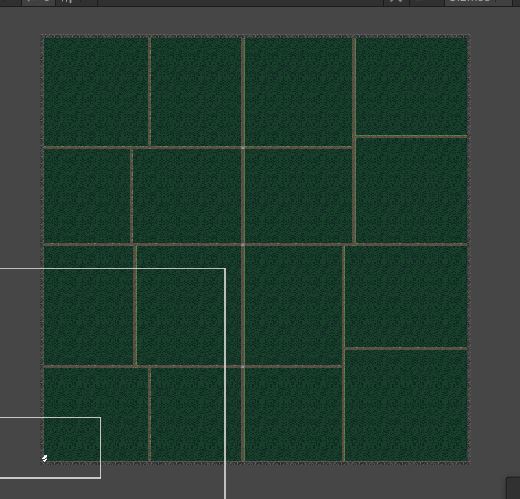
**제 4 장 실험 결과**

**제 1 절 실험 환경**

Unity 2d를 이용하여 실험하였다.

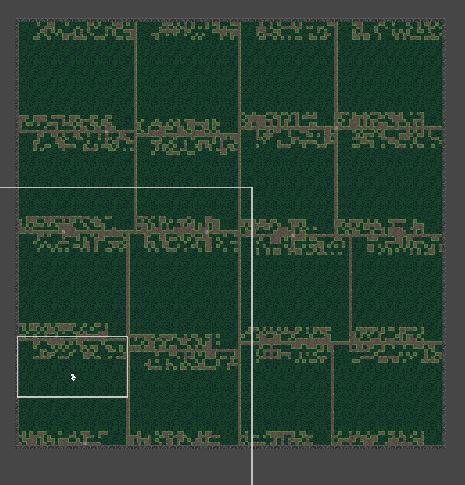
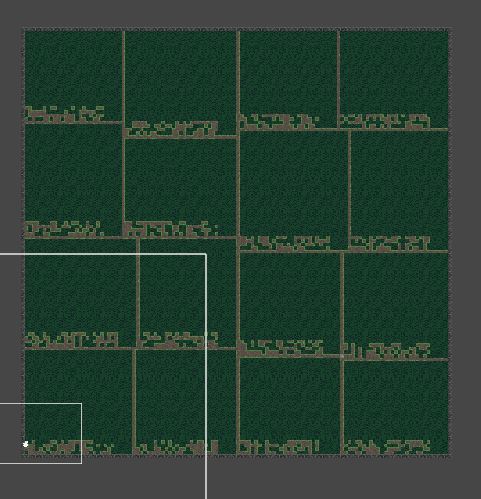
**제 2 절 실험 결과**

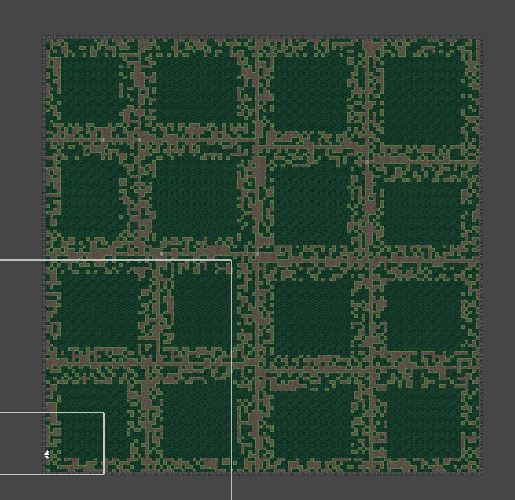
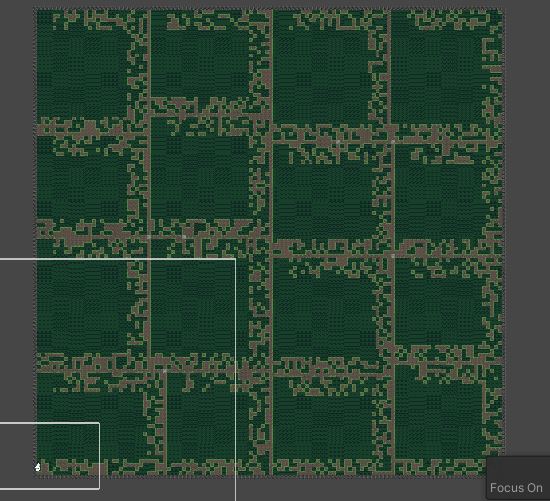
BSP로 구역을 나눈뒤 각 컴포넌트에 CA를 적용하였다. BSP의 탈출 조건은 컴포넌트의 크기가 1200px보다 작을 때로 하였다.



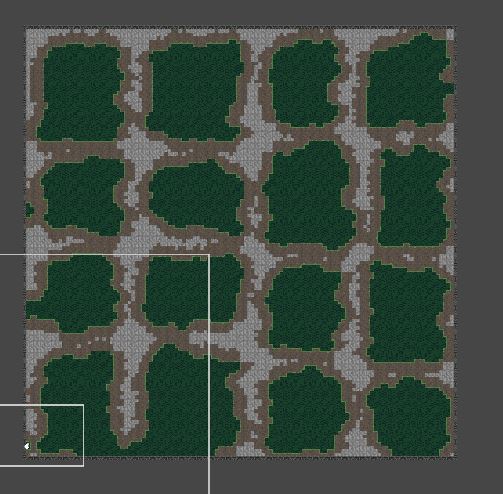
Bsp로 위 사진과 같이 컴포넌트들을 생성해준뒤 각 컴포넌트에서의 중심에서 멀수록 노이즈의 비율이 많아지도록 구현하여 주었다. 색칠할 때 둘레의 모양은 직사각형에서 직사각형이 빠진 모양이라서 어떻게 나눠야 효율적이지 감이 오지 않았었다. 왼쪽 하단 가로, 오른쪽 상단 가로, 오른쪽 하단 세로, 왼쪽 상단 세로로 사분할하면 예외 처리

없이 쉽게 해결할 수 있었다.



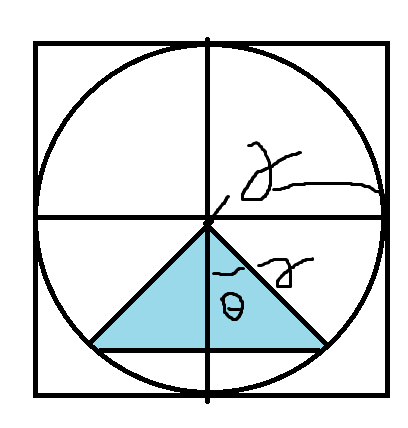


위 사진들이 언급한 순서대로 위치한 곳에 노이즈들을 생성한 결과다.



우리가 원했던 모양은 좀 더 가지각색이며 원에 가까운 형태를 원했는데, 사각형에 가까운 형태가 자꾸 나왔다. 그래서 중점의 거리에 따른 노이즈들의 생성 비율이 바뀌게 연구를 진행 하였다.

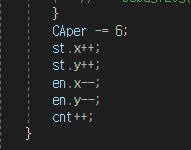
학교에서 배운 삼각비를 이용하여 거리에 따른 비율을 쉽게 유추 할 수 있다.



위 그림에서 길이가 2r인 정사각형이 있고, 이에 내접하면서 지름이 2r인 원이 있다.

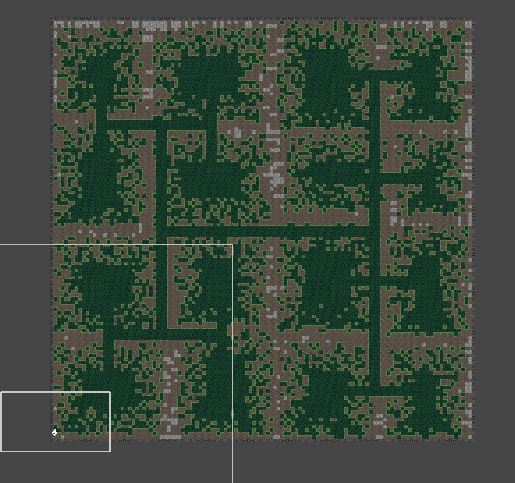
원에서 임의의 현은 r 거리에 있으며, 이 현의 길이는 2\* r이므로 중점에서 거리에 있을 때 둘레에 노이즈 생성 퍼센트는 (1-)\*100%만큼 생성해주면된다.

하지만, 구현 능력 부족으로 삼각함수를 구현하지 못해서 중점에서 거리가 1씩 가까워 질수록 노이즈 생성 퍼센트를 6퍼센트씩 낮췄고, 둘레에 색칠되는 길이가 8px 이 될 때까지 반복하였다.

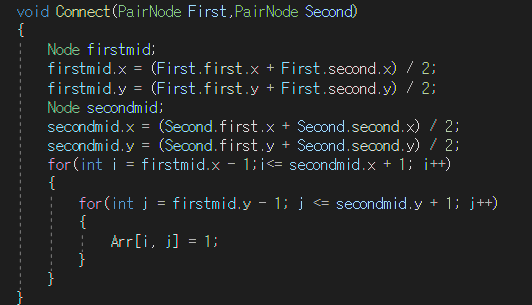




에서 거리에 따른 등차수열로 노이즈들을 생성해주었다면, CA를 통해 노이즈들을 처리해준다.



각 컴포넌트들을 연결 시켜주는 방법은 bsp tree에서의 성질을 이용하였다. 같은 부모노드를 가지고 있는 컴포넌트들 끼리 연결 시켜주면, 모든 컴포넌트들이 연결 된다.



**제 5 장 최 적 화**

**제 1 절 알고리즘 구현**

노이즈들을 생성할 때 랜덤으로 생성하는데, 원래 색칠되지 않은 노드를 색칠할때까지 while문을 이용하여 작성하였더니, 최악의 경우에 무한에 수렴하는 시간 복잡도가 나왔다. 그래서 색칠할 구역들에 랜덤 값을 부여한뒤, 랜덤 값에 따라 정렬하면 시간복잡도를 O(NlogN)까지 줄일 수 있다.

**제 2 절 성 능**

BSP tree의 시간 복잡도는 하한선은 O(N^3)이 나오고, 공간 복잡도의 하한선은 O(N^2)이다. 컴포넌트들의 전체적인 위치는 매 순간 비슷하지만, 모양은 항상 다르다. Cave 형태의 던전게임에서의 활용도가 무궁무진하다.

**참고 문헌**

<https://www.youtube.com/watch?v=Hebi641Ph7c&feature=emb_logo>

<http://1st.gamecodi.com/board/zboard.php?id=GAMECODI_Talkdev&no=3763>

<https://ko.wikipedia.org/wiki/%EC%84%B8%ED%8F%AC_%EC%9E%90%EB%8F%99%EC%9E%90>