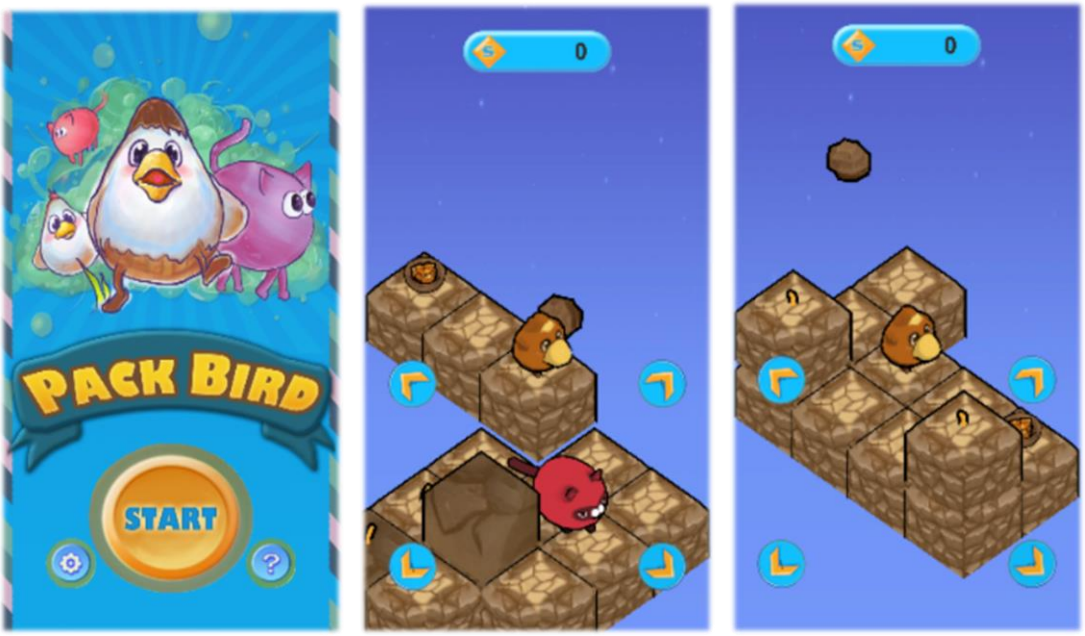
PackBird



개발 기간 2015.04 ~ 2015.06

개발 인원 박민후, 황인원, 이유리, 이은정

개발 스펙 Unity [C#]

개발 내역
1. 그래프 구현 및 길찾기 구현 [A*]
2. 캐릭터 움직임 구현
3. 유니티 에디터를 확장한 맵 에디터 구현

요 약 팀 프로젝트 수업 때 개발했던 게임입니다. 팩맨과 같이 돌아다니면서 지렁이를 먹고 일정 점수 이상을 획득하면 클리어합니다. 고양이를 만나거나 떨어지는 돌과 부딪히면 게임오버가 됩니다.

소스 코드 https://github.com/InwonHwang/PackBird

게임 영상 https://www.youtube.com/watch?v=I3rcVQ_YmoM

Priority Queue

AStar알고리즘을 구현할 때 유니티 C#에서 우선순위큐를 지원해주지 않아서 직접 구현하여 사용하였습니다. 힙자료구조 사용하여 구현하였습니다.

```
public class PriorityQueue<T> where T : new()
{
    List<T> _list;

    public delegate bool CompareMethod<T1>(T1 a, T1 b);

    CompareMethod<T> _compare;

    public PriorityQueue(CompareMethod<T> method)
    {
        _list = new List<T>();
        _list.Add(default(T));
        _compare = method;
    }

    public void Push(T value)
    {
        _list.Add(value);

        heapify(_list.Count - 1);
    }

    public T Top()
    {
        if (Empty())
            return _list[0];

        return _list[1];
    }

    public bool Empty()
    {
```

<PriorityQueue>

AStar

고양이가 새와 같은 높이에 있을 때 고양이가 새를 쫓아가는 기능을 구현하였습니다.

```
public class Graph
{
    class Node
    {
        public float heuristic;
        public float goal;

        public Box value;
        public Node parent;
    }

    List<Box> _boxes = new List<Box>();
```

<Graph>

```
public List<Vector3> GetPath(Vector3 start, Vector3 end)
{
    PriorityQueue<Node> pq = new PriorityQueue<Node>((Node a, Node b) => {
        return ((a.goal + a.heuristic) > (b.goal + b.heuristic)) ? true : false;
    });
    List<Box> visit = new List<Box>();

    Node startNode = new Node { heuristic = Vector3.Distance(start, end),
        goal = 0,
        value = Find(start + new Vector3(0, -1.0f, 0)),
        parent = null
    };
    Node cur = null;
    Node target = null;

    pq.Push(startNode);
    visit.Add(startNode.value);

    while (!pq.Empty())
    {
        cur = pq.Top();
        pq.Pop();

        if (cur.value && cur.value.position == end) target = cur;

        for(int i = 0; i < 4; ++i)
        {
            if (cur.value == null) break;

            if (cur.value[i] && !visit.Contains(cur.value[i]) &&
                cur.value[i].position.y == cur.value.position.y)
            {
                Node child = new Node
                {
                    heuristic = Vector3.Distance(cur.value[i].position, end),
                    goal = cur.goal + Vector3.Distance(cur.value[i].position, cur.value.position),
                    value = cur.value[i],
                    parent = cur
                };

                pq.Push(child);
                visit.Add(child.value);
            }
        }
    }
```

<AStar>