18/05/2015

Dromund Kaas Game Specification

I. Engine

Main module; handles game logic and ties all the other modules in.

A. Representation of internal data:

HashSet<EntityType> EntityTypes;
// to be loaded when the program starts - all entities parsed from file "EntityTypes.dk". See EntityType - Encoding for information on format. (use Regex)
// also use a HashSet for EnemyTypes and PlayerTypes;
List<Entity> Entities;
// to be filled dynamically as entities are generated - one Player Entity and multiple Enemy Entities, based on the EntityTypes.dk
Queue<bool[]> EnemyBullets;
// matrix of enemy bullets; 1 = bullet; 0 = no bullet; Roll down 1 row every game cycle
Queue<bool[]> PlayerBullets;
// matrix of player bullets; 1 = bullet; 0 = no bullet; Roll up 1 row every game cycle
int CycleCounter;
//to count cycles

Functions:
static void LoadEntityTypes();
//load all entity types upon program start, from the file EntityTypes.dk. Assign colors to entities.
static void RollUp(Queue<bool[]>);
//roll queue up
static void RollDown(Queue<bool[]>);
//roll queue down

…

B. Processing of data:
    a. Initialize variables
    1. Load new cycle - increment CycleCounter;
    2. Progress bullets
        a. move enemy bullets down
        b. move player bullets up
    3. Match bullets
        a. enemy bullets in same space as player?
            i. yes: decrease life, delete bullet
                1. Enemy life 0? delete entity, increase player kill counter
        b. player bullets in same space as enemy?
            i. yes: decrease life, delete bullet
                1. Player life 0? Game Over
    4. Progress entities
        a. enemies move along their routes
        b. player moves up, down, left, right, according to last pressed key
        c. Ensure there are no collisions!!
    5. Load new bullets
        a. load enemy bullets into enemy bullet matrix (set to 1 or true)
            i. Regex catching ‘@’
        b. load player bullets into player bullet matrix (set to 1 or t)
            i. Regex catching ‘$’

C. Output
    a. Implement anti-flicker solution
    b. Draw all backgrounds
    c. Draw all bullets
    d. Draw all entities
        i. set cursor, draw with color
    e. Draw stat counter at the bottom
        i. Player lives, player kills

II. Entities

Module to describe entities within the game - Player, Enemies.

Utility struct: struct Point {...} //position in 2d space
Point parameters:
```
int     X, Y;  //coordinates
```

- **Main class: class Entity {...}**

Entity parameters:
```
int     Life;   //how many life points the entity has by default
Point   Location;       //the location of the Entity
int     Step;   //the current movement step
```

- **class EntityType {...}   //type of entity; one instance per type**

Implements: IComparable;
//(implement it yourself; compare by Name)
EntityType parameters:
```
string  Name; //the entity name; every entity type has a unique name (Player, Enemy1,...)

char[,] Sprite;//the image of the EntityType; what will be drawn
        // E.g.:
        // { '(', '=', '0', '=', ')' }
        // { ' ', ' ', 'V', ' ', ' ' }

int     MaxLife;        //the maximum life a type of entities has by default
string  Movement;   //a string of movement instructions the entity takes by default
        // allowed characters: 'u' - up;
        //                     'd' - down;
        //                     'l' = left;
        //                     'r' = right;
        // e.g. Movement = "dllrr", to be repeated when the end of the string is reached
```

Constructor:
```
EntityType(string N, char[,] S, int M, string Mov)
{
        this.Name = N;
        this.Sprite = S;
        this.MaxLife = M;
        this.Movement = Mov;
}
// ...
```

- **Encoding**

File: “EntityTypes.dk”

Format:
//!! player names always start with “player”. Everyone else is an enemy. Boss type names start with “boss”
//!! enemy blasters are ‘@’; player blasters are ‘$’

```
>>>>>
name=”playerLuke”;
life=”3”;
    A
|$ | | $|
|==|_|==|
>>>>>
name="Skalichka";
life=”5”;
movement=”dllrr”;
 /0\
/_@_\
>>>>>
```

III. Async

- Asynchronous Music Module.
    - option to stop music

1. Intro music - v
2. Battle music - ?
3. Boss music - ?

- Asynchronous last button pressed tracking

IV. Intro/Outro

Intro/Outro art module.

1. Add color to intro