PA8

Inspiration:

Zelda

Pokemon Red

Megaman

Movement:

Keyboard:

~~Left~~

~~Right~~

~~Up~~

~~Down~~

~~Forward & Back?~~

Input:

~~Arrow keys~~

W-A-S-D, not arrow keys, since we’re going to use the mouse and

Most people are right handed, those keys won’t work.

E, interact(open chest, talk, open door, ect)

Targeting:

Mouse controlled

Left click = object in left hand ~~fire / use~~

Right click = object in right hand

Attack:

At will of player for player

Determined by proximity by enemy (Need a proximity algorithm (same as collision just

With a wider area))

Need to determine weapon range – Done!

Sword = 8 pixels

Enemy = contact / collision

Defense/shield:

Define a button for defense – use shield

Environment:

Platforms?

Bridges?

Ice (add terrain class to determine movement)

Several screens in size

Enemies:

Will inherit from Actor class

Application class will control movement?

Simple moves

Position trigger

Some will move towards player

Pseudocode:

// check enemy position compared to player position

// if the difference in x is greater than difference in y

// move towards player in x dimension

// otherwise, move in y direction (this will be the direction if

// they are equal as well)

// this will have to take place after player evaluates for collision

// and if there is a collision, it will be handled as such

// before move is called, enemy object will need to use the

// proximity algorithm to determine if it should move or attack

If (enemy.x – player.x < 0)

{

AI.x = AI.x \* -1; // if negative, make positive (absolute value)

}

If (enemy.y – player.y < 0)

{

AI.y = AI.y \* -1; // if negative, make positive (absolute value)

}

// if the difference in x distance is greater than the

// difference in y distance, move in the x direction

// otherwise, move in y

If (AI.x > AI.y)

{

If (enemy.x – player.x > 0) // we are to the players right

{

Move one space in –x // move left

}

Else

{

Move one space in +x // move right if we are not to the

// players right

}

}

Else

{

If (enemy.y – player.y > 0) // if we are to players down

{

Move one space in –y // move up

}

Else

{

Move one space in +y // otherwise, move down

}

}

Boss:

Multiple simple moves

Will move towards player

Programming:

Resources directory:

Contains style sheets for actor(s), menu, world/background

Classes:

Object:

Defines objects position,

inputComponent:

Controls all inputs

graphicsComponent:

Will control bitmaps displays, which view of the sprite sheet is shown

Actor:

Actor class, define actor elements (formerly Character class)

Hero:

Inherits from Actor class, attackSword and attackFireball moved here

Enemy:

Inherits from Actor class

Background:

Control the background, move all the tilesets for the background here?

Application:

Wrapper, control the game from this class

TileMap:

Tile map, contains tile sheet data

Room:

Area will be divided into “rooms” that will control n number of elements

As the player moves about, they will transition from one to the next

Seamlessly so as not to detract from game play

Room will check for collisions and call hero.move to move player if no

Collision

Weapon:

Was an idea to be its own class, this will be moved to the Actor class

World:

Control the rooms

How Marcus envisions things:

Application:

* Room
  + number
  + Set size
* Background
  + Load blank background of room size
  + Load tile-sets to fill room
* Interactive elements
  + Obstacles
    - X, Y, Width, Height on background
    - Load images
  + Fireball spell
    - X, Y on background
  + Equipment Upgrade(s)
    - X, Y on background
* Opponents
  + Random X, Y on background
  + Default Move pattern for some
  + Sit and wait for hero to enter range for some
* Hero
  + Hero moves through game
    - First room, begin in center of screen
  + As move to new room, save room elements, repeat all above except that hero position is relative to its position from the previous screen and not centered in the new window
* View
  + Start with middle of first room
    - Moves as hero moves
      * If viewable window size is 800X600, as hero position gets within 200 of edge of view, move view with hero
        + Experimented with changing from map.move to character.move and camera now follows character in characterviews.
* Opponents have set ranges
* Some opponents have proximity detection, when the hero enters their proximity, they will move towards hero.
  + Two different proximity ranges
    - 1. Move-to-hero
    - 2. Attack
* Some opponents have set move patterns and move along them regardless of hero position
* Every move will have collision evaluation for all opponents and hero
* Obstacles are objects that prevent other objects from occupying the same space
* Fireball spell and upgrades are objects that will interact with hero
  + How to interact?
    - Option 1: solid, walk up to it, interact and obtain
      * space is freed
    - Option 2: not solid, walk through it to interact