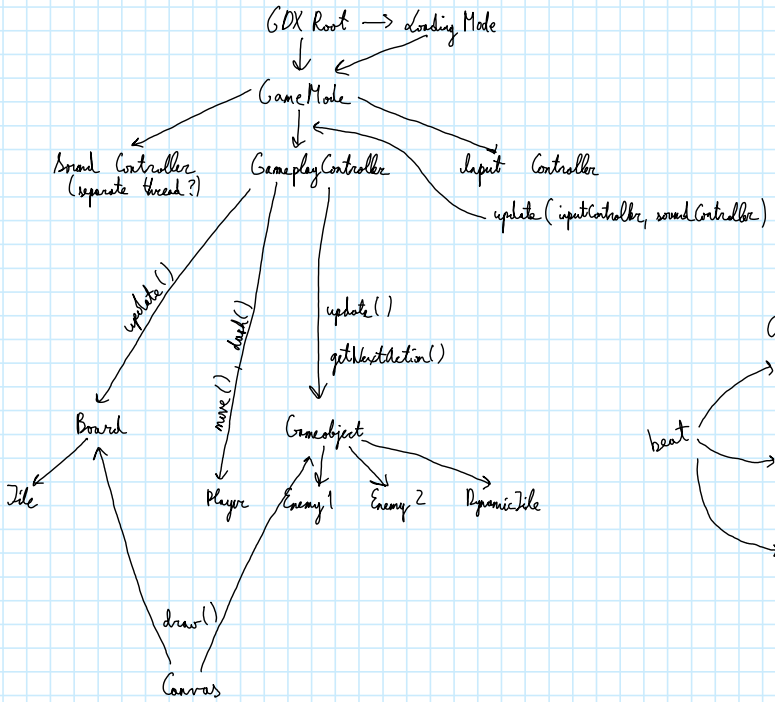
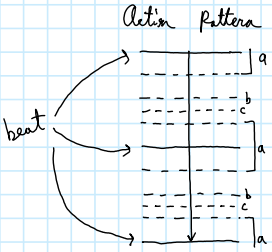
More Architecture (Technical Prototype)

Thursday, February 26, 2015 12:54 PM





a → check for player actions
if appropriate action happened, resolve player/NPCs

b → if player still hasn't moved, resolve player/NPCs appropriately

c → refresh beat state (I.e. player moved, player invulnerable, etc.)

Canvas / Models → Kylee

Input Controller / Player Controller / Character and Enemy update/move calls → Austin

Gameplay Controller / Main Game loop → Charlie

RhythmController → Gagik

Things to implement:

Kylee: Canvas (just copy over old canvas)

Abstract GameObject
- Player
- Enemy (Melee)
- Enemy (Ranged)
- Dynamic Tile
- Board / Tile

Extends GameObject

Bullet

come up with names?

GameObject implements (abstract)
- draw()
- isAlive()
- isActive
- update()
- getPosition()

Each subclass of GameObject contains bytecode that correspond to its own actions
(we could also use enums, but they're not as robust)

E.g. player:

```
public static final int PLAYER_MOVE_RIGHT = 0x0000;
    :
public static final int PLAER_DASH_UP = 0x0010;
    :
etc.
```

Remember to set correct object state (for animation) with appropriate bytecode commands.

Only put code in update() that we expect to happen every frame (i.e. update sprite, tick up any counters).

Board stores an nxm array of tiles. Implements methods that return the tile state for a given position

Enemy classes should store an array of bytecode commands for their cycling actions

At each act(), if the expected command happened, go to the next one. If not, keep trying the same one.

Heirtin: InputController (abstract)
PlayerController
AIController
Certain methods in GameplayController

InputController implements getAction(), which returns bytecode.

May want to give AIController a subclass for each enemy type, or just use an enum in switch statement

OR might figure out a way to make it invariant b/w enemies (might agree with Kyber to implement a getNextAction in enemy models)