Meeting Agenda

Date: 2015-05-19

Facilitator: Jesper Olsson

Participants: Fredrik Bengtsson, Jesper Olsson, Anton Strandman, Ulrika Uddeborg

Objectives:

§1 Decide what to do until next meeting.

Reports:

- Models - Anton
    - Dragon and ogre
        - Tried it out, not looking to good...
- No instakills
    - FirstCollisions
        - done, did not use FirstCollisions but methods in model.
- Profilechange - Anton
    - Menu is setup and it’s possible to select a profile.
    - Still missing add and remove profile functionality.
- Box
    - not yet done
- Save/load data (profile) - Anton
    - Done

    - Profiles are saved upon change (although will need to be saved manually when profile is created)
    - All saved profile files are loaded on startup
- No more flying
    - Fixed. Bug was in NodeFactory.
- Death/respawn
    - done, healthcontrol questionable
- Cleanup
    - Not done, need refactoring.
- Relocate KeyBindings
    - Is private inner class in InputManagerWrapper, which provides relevant methods via delegation.

Other:

- Fixed some null pointer in profile.

Discussion items:

From the paragraphs above:

1

Other:

- In player, there are two collide(). The one from the interface is not used.
    - might not need an abstractNPC, always take one damage from all collsions?
- Questions about
    - INode
        - See comments in class
    - Abstract character
        - Should there be a default behaviour when colliding?
    - Player

        - Has methods setIsDamaged() and getIsDamged(). Abstract to inteface IDamageable().
        - Health is changed in collide.
- Factories
    - 1. Scan nodes in level and add model to the node, depending on node data.
        - The model have access to all data types and control types needed.
            - Example: Player contain a “MoveControl”, and in the constructor of the player it requests a MoveControl but is provided one of the subclasses: PlayerMoveControl. An NPC will require the same information but will be provided an NPCMoveControl.
                - Both controls will be subclasses to MoveControl.
                - The getMoveControl() will be provided from the AbstractCharacter class, thus abstraction can be handled on the abstractCharacter level for all Character type nodes.
    - 2. Send the node to painter
    - 3. Painter check for model class abstractionlevels:
        - If instance of character: Load character specific data
            - If instance of NPC: Load NPC specific data
        - Else If instance of inanimate load inanimate data.
        - There should be NO data types that only 1 class. Everything should be on abstraction level.
        - Which control is received when running the abstract methods depend on the class itself

Outcomes and assignments:

- Clean up - Jesper
    - Factories
    - Look through the rest of the classes.
- RAD - Anton
- SDD - Anton
- Pause/unpause - Fredrik
- AI - Ulrika
- End level - Fredrik
- Finalize profile menu
- Profile update inputmanager - Ulrika
- textures and models - Fredrik

Wrap up:

Next meeting: 2015-05-20