Домашняя работа 6. Архитектура Roguelike

28.02.2022

Дедлайн: 10:00 14.03.2022
Баллов: 10

Roguelike — это довольно популярный жанр компьютерных игр, назван в честь игры Rogue, 1980 года выхода. Характеризуется:

- простой тайловой или консольной графикой (современные игры roguelike-игры иногда имеют вид сбоку (Dead Cells, Noita) или даже 3D (Risk of Rain 2), но (псевдо-)консольных и тайловых тоже полно (Caves of Qud, Cogmind, например));
- активным использованием случайной генерации;
- перманентной смертью персонажа и невозможностью загрузить предыдущее сохранение (не чтобы позлить игрока, а чтобы дать ему возможность попробовать разных персонажей и разные пути развития);
- чрезвычайно развитым набором игровых правил (чем эти игры нам и интересны, модель их предметной области может быть как очень простой, так и невероятно сложной);
- высокой свободой действий персонажа (так называемые «игры-песочницы»)

Классические примеры:

- https://en.wikipedia.org/wiki/NetHack
- https://en.wikipedia.org/wiki/Angband_(video_game)
- https://en.wikipedia.org/wiki/Ancient_Domains_of_Mystery

Вашей задачей будет в командах по три-четыре человека разработать архитектуру такой компьютерной игры.

При этом должны быть выполнены следующие функциональные требования:

- персонаж игрока, способный перемещаться по карте, управляемый с клавиатуры;
  - карта обычно генерируется, но для некоторых уровней грузится из файла;
  - характеристики — здоровье, сила атаки и т.д.;
- у персонажа есть инвентарь, состоящий из вещей, которые он носит с собой;
  - вещи из инвентаря можно надеть и снять, надетые вещи влияют на характеристики персонажа;
  - вещи изначально находятся на карте, их можно поднять, чтобы добавить в инвентарь;
  - снятые вещи находятся в инвентаре, их можно надеть в дальнейшем;
- консольная графика, традиционная для этого жанра игр.

В данном задании требуется разделиться на команды (можно не так, как в CLI) и написать архитектурное описание Roguelike, как обычно пишутся design document-ы:

- общие сведения о системе;
- ключевые требования (architectural drivers);
- роли и случаи использования;
  - описание типичного пользователя, как, надеюсь, было в курсе по SE;
- композиция (диаграмма компонентов и её текстовое описание);
- логическая структура (диаграмма классов и её текстовое описание; может быть по одной диаграмме классов на компонент, но все связи должны быть прослеживаемы — то есть нужны и классы из других компонентов с полностью квалифицированными именами, но без атрибутов/операций, если на них ссылаетесь);
- взаимодействия и состояния (диаграммы последовательностей и конечных автоматов и их текстовое описание — даже если для вашей архитектуры они не очень полезны, нарисуйте, потренироваться).

На что обратить внимание:

- на разделение системы на компоненты — решения вида «большой клубок классов» будут оценены очень низко;
  - выберите и используйте какой-либо архитектурный стиль, явно укажите его в описании;
- на прослеживаемость потока управления — должно быть понятно, с какого места запускается программа, кто кому передаёт управление;
- что все имеют необходимые для своей работы данные — например, пользовательский интерфейс знает про карту;
- на баланс детальности и читаемости диаграммы — она должна быть достаточно детальна, чтобы при реализации не требовалось принимать серьёзных архитектурных решений, но при этом обозрима. Например, старательно выписывать все методы классов и все поля не надо, но ключевые методы и поля всё-таки нужны;
- на согласованность диаграмм — на диаграммах последовательностей не должно быть классов, которых нет на диаграмме классов, например;
- на следование синтаксису UML.

Эту архитектуру в следующих заданиях надо будет реализовать (в составе тех же команд, в которых она проектировалась). Далее каждую неделю будут появляться новые требования, поэтому проектируйте архитектуру расширяемой и по возможности гибкой.