# Личный вклад

1. Классы Character, Hero и NPC

Главными действующими лицами игры являются персонажи, поэтому первоочередной задачей было создать класс для игровых и неигровых типов.

Персонаж должен иметь:

Инвентарь, методы взаимодействия с инвентарём;

Экземпляр класса Skin, отвечающий за визуализацию и физические взаимодействия;

Поле фракции;

Поля для оружия  и брони, указывающие на соответствующие вещи в инвентаре (все персонажи потенциально могут биться);

Поле SEACIL (Strength, Endurance, Accuracy, Charisma, Intelligence, Luck) – характеристики, влияющие на ролевую составляющую для данного персонажа.

Класс Character наследуется от cocos.layer.ScrollableLayer, чтобы на него можно было установить Skin, а его самого на карту. Конструктор принимает имя (по которому создаётся Skin), SEACIL, фракцию, mover (специальный объект, отвечающий за движение данного персонажа) и позицию на карте на момент создания. Для этого класса я написал:

- Методы взаимодействия с оружием (достать\спрятать, перезарядить, атаковать, взять в руки\положить в инвентарь).
- Методы взаимодействия с бронёй (надеть\снять).
- Методы взаимодействия с внешним миром (получить урон, подобрать предмет, выбросить предмет, положить предмет в сундук).
- Методы взаимодействия с собой (использовать предмет, изменить характеристику).

Класс Hero наследуется от Character и дополнительно содержит обработчики мыши и клавиатуры.
Класс NPC наследуется от Character и дополнительно содержит искусственный интеллект, характеристики npc загружаются из файла.

2.  Интерфейс и меню

Интерфейс предоставляет информацию о состоянии главного героя и позволяет взаимодействовать с ним.

Главный интерфейс представлен классом interface. Он наследуется от cocos.layer.MultiplexLayer, чтобы можно было активировать только один вид интерфейса в один момент времени и переключаться между ними. Главный интерфейс состоит из следующих частей: просмотр своего инвентаря (visual_inventory), информация о здоровье, броне, выносливости, текущем оружии, и интерфейс обмена с сундуком (или другим персонажем). Interface обрабатывает нажатия клавиш и по ним запускает нужный элемент интерфейса (на Q инвентарь, на E или ЛКМ обмен, если рядом есть сундук, меню игры на ESC). Стоит заметить, что за исключением меню игры, другие элементы не ставят игру на паузу, это сделано специально.

Элементы интерфейса, содержащие инвентари наследуются от класса BasicVisulInventory, предоставляющего простой просмотр инвентаря, без взаимодействия. Элементы просмотра своего инвентаря и обмена добавляют на него нужные кнопки (выбрасывания предмета, экипирования и т.д.).

3. Карта
Для создания карт мы выбрали Tiled Map Editor, потому что это первый редактор двумерных карт, который мы попробовали, он показался нам удобным и мы не стали искать дальше.
Для карты я нарисовал необходимые тайлы и создал наборы тайлов в данном редакторе, затем создал саму карту. На карте я выделил отдельные слои для стен, пола, декораций и объектов над картой. Есть отдельный слой видимых объектов, обладающих помимо визуальной составляющей физическими свойствами, и невидимый слой с физическими моделями объектов, размером меньше клетки. Последнее решение пришлось использовать из-за ограничений cocos2d.
Это и составило основную трудность: выявлять, что cocos не умеет обрабатывать. Сюда вошли: повороты тайлов, тайлы разных размеров (у каждой карты есть фиксированный допустимый размер тайла, например 32x32, от которого нельзя отступать), круглые физические модели (если на слое объектов поставить круглый примитив, то в кокосе он обведётся квадратом).

4. Инвентарь и сундуки
Инвентарь содержит 4 массива: обычных item, оружия, брони и используемых объектов. В нём находятся методы получения предметов, выбрасывания предметов, добавления предметов. Все эти методы принимают имя предмета.
Сундук, который наследуется от cocos.layer.ScrollableLayer, содержит инвентарь, спрайт, отвечающий за внешний вид, и физическую модель. В него можно класть предметы, и доставать их из него. Этот класс содержит статический словарь всех лежачих объектов, где ключ это хэш от самого сундука (hash(self)).
При коллизии с сундуком главный герой сохраняет себе его уникальный ключ и уже в обработчике клавиш и мыши (поднимаемый объект можно поднять либо на E, либо по клику мыши) он вызывает у интерфейса метод обмена с данным сундуком, полученным по ключу.
Обработка обмена с сундуком происходит аналогично взаимодействию с PickableObject (см. дальше).

5. Поднимаемые предметы
Класс PickableObject содержит спрайт, имя предмета, хранящегося под данным спрайтом, количество предметов и дополнительные параметры (например количество патронов у оружия). Он также содержит физическую модель, при коллизии с которой главный герой может поднять предмет.
Этот класс содержит статический словарь всех лежачих объектов. При коллизии с объектом главный герой сохраняет себе его уникальный ключ и уже в обработчике клавиш и мыши (поднимаемый объект можно поднять либо на E, либо по клику мыши, отличие в том, что при клике поднимается конкретный объект, а при нажатии E первый из нескольких рядом лежащих) он подбирает нужный объект, обращаясь к словарю по ключу, и вызывает у него метод удаления с карты.
Основной сложностью данного задания было придумать, как дать объекту уникальное имя. Для этого я решил брать хэш от объекта PickableObject (hash(self)).