Отчёт

# Декомпозиция

Эту декомпозицию я делал до того, как начал делать игру. В реальности начал с геймплея, потом пауза, потом конец игры и после главное меню

## Главное меню

* Старт
  + кнопка старт
  + переход к игре
* Очки
  + окно с максимально набранным счётом
  + кнопка закрыть
  + система хранения очков
    - сохранение результата
    - загрузка результата
* Настройки - Отображается окно настроек, в котором есть кнопка включения/отключения звука и кнопка “Закрыть”
  + окно
  + кнопка включения выключения звука
  + включение выключение звука
  + сохранение настроек

## Игровой процесс

На экране игрового процесса (после нажатия кнопки “Старт”) - игрок должен видеть “Очки” и на экране должна быть кнопка

* Отображение очков
  + текст в углу
  + система начисления очков
* Система ввода
  + UI панель
  + обработка клика по ней
  + событие для подписки
* Птичка
  + движение
    - движение вперёд
    - сила притяжения
    - подскакивание
  + упасть
* Следование камеры за птичкой
  + движение камеры (только по оси x)
  + задний фон на ней
* Генерация уровня
  + параметры генерации
  + генерация преграды когда птичка прошла следующее препятствие генерация уровня

## Пауза

* Пауза
  + кнопка паузы
  + остановка / возобновление времени
  + окно
  + кнопка продолжить
  + кнопка выход в меню

## Поражение

* окно проигрыша
  + окно
  + текущие набранные очки
  + максимально набранные очки
  + кнопка выхода в меню

# Затраченное время

Точно сказать не могу, так как много времени потратил на изучение zenject-а так как раньше с ним никогда не работал, а также часть времени ушла на то, чтобы построить архитектуру, как в примере, который вы показали <https://habr.com/ru/post/343728/#comment_10549160>. Если очень приблизительно, то часа 24, но так как я разобрался, следующий проект с такой архитектурой сделаю быстрее

# Чем вы руководствовались при выборе архитектуры для игры?

Своим предыдущим опытом и примером, который вы прислали)

# Как вы построили работу с UI, почему именно так?

Как и со всем остальным, monoBehaviour отвечает за простейшие функции, связанные с видом, а обычные классы им управляют и общаются друг с другом через event-ы

# Какими паттернами объектно ориентированного дизайна вы воспользовались при разработке проекта?

observer (ивенты), фабрика (для создания препятствий через zenject-овский instanciate)

# Соблюдаются ли в вашем коде, принципы SOLID, если да, какие из них?

Соблюдаются. SRP, OCP. Остальные я не стал реализовывать так как слишком маленькая игра, интерфейсы и абстрактные классы для такой были бы избыточными

# Какие изъяны/технический долг есть в этом проекте с точки зрения разработки?

Можно бы было раскидать инсталлеры с одного на несколько разных для лучшей читаемости. Некоторые классы лучше загружать на весь проект.

# Если бы это был коммерческий проект с гораздо большим бюджетом времени на разработку, что бы вы сделали по другому?

см пункт выше + больше бы делал упор на абстракцию, выделил бы интерфейсы, сделал бы красивые анимации

# Что вызвало у вас наибольшие сложности при разработке этого задания?

Некоторые вопросы по zenject-у которые не гуглились, пришлось спрашивать в чате дискорда по нему)

# Если есть что-то важное, что вы хотели бы сказать про это задание, можете написать в свободной форме.

Очень рад, что попробовал контейнеры, очень удобная штука, которой раньше не хватало.

Задачу саму игру сделал полностью через контейнеры, на главное меню сил вообще не осталось, поэтому очень надеюсь, что ничего, что там нет вкладок очки и настройки, они делаются аналогично окнам поражения и паузы. (контейнеры в нём тоже не использовал там одна кнопка «начать игру»)

Upd: управление удобнее на пк мышкой чем на андроиде так как тестировал в основном на пк

Upd2: Сам проект находится в папке flappy bird в корне репозитория

Upd3: Не сделал удаление препятствий (не хватило сил/не успел), делается очень легко через коллайдер сзади, они будут на него наезжать и через скрипт того коллайдера удаляться. Второй вариант удалять через корутину по времени