Visual Studio как фронт-энд для графического движка

Александр Кухаренко

@lastshilling

0f0x64@gmail.com

Александр Кухаренко

@lastshilling

Биография

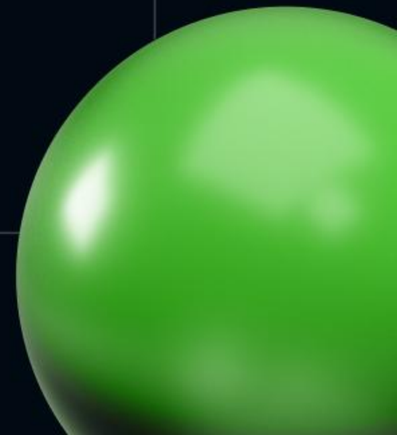
Занимаюсь компьютерной графикой с 1997 года. Работал в различных компаниях (Wargaming, Wanna, Nevosoft и др.) на позиции рендеринг-инженера, также занимался техартом, дизайном спецэффектов, настраивал пайплайны работы для 3д-артистов.

Люблю оптимизацию и красивые алгоритмы, поэтому сделал много работ для фестивалей компьютерного искусства (demoscene).

Сейчас поднимаю стартап - студию разработки игр.

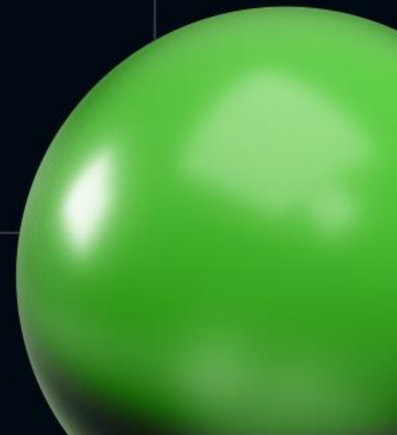
План доклада

- Концепции UI и их проблемы



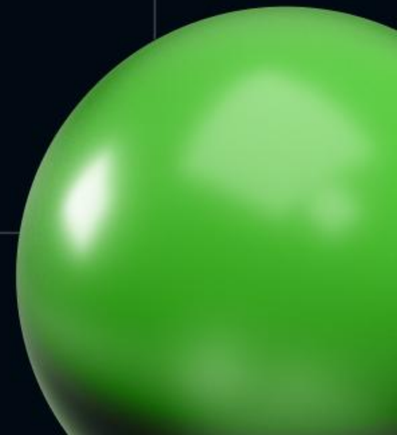
План доклада

- Концепции UI и их проблемы
- Требования к UI



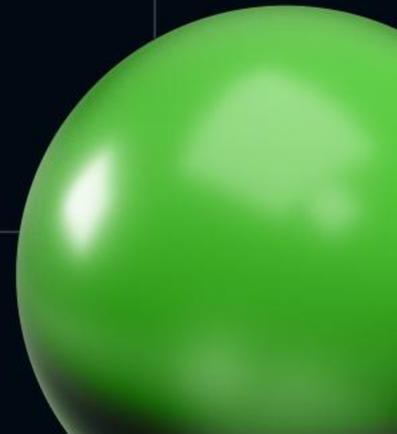
План доклада

- Концепции UI и их проблемы
- Требования к UI
- Демонстрация решения на примере рабочего проекта



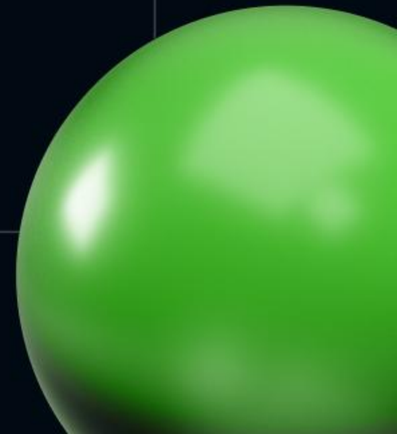
План доклада

- Концепции UI и их проблемы
- Требования к UI
- Демонстрация решения на примере рабочего проекта
- Архитектура решения
  - Интеграция шейдеров в C++ движок
  - C++. Отклик в реальном времени на изменение кода
  - Апгрейд UI Visual Studio



План доклада

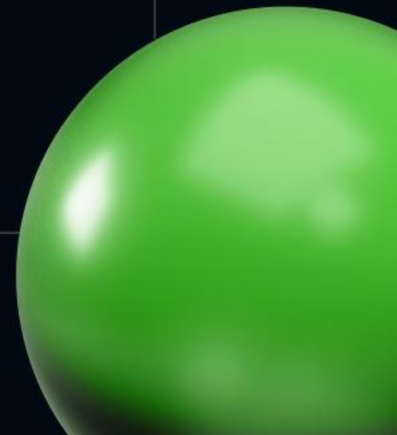
- Концепции UI и их проблемы
- Требования к UI
- Демонстрация решения на примере рабочего проекта
- Архитектура решения
  - Интеграция шейдеров в C++ движок
  - C++. Отклик в реальном времени на изменение кода
  - Апгрейд UI Visual Studio
- Преимущества и недостатки подхода



Порог входа в UI

1985: примитивные редакторы, текстовые конфиги

CG-художники работают в контр-интуитивной для них среде



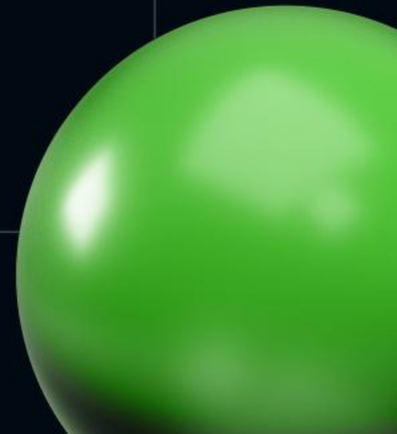
Порог входа в UI

1985: примитивные редакторы, текстовые конфиги

CG-художники работают в контр-интуитивной для них среде

2005: стеки, ноды, слайдеры, чекбоксы, меню

среда нативно удобна для CG-художников, но ее сложность быстро растет



Порог входа в UI

1985: примитивные редакторы, текстовые конфиги

CG-художники работают в контр-интуитивной для них среде

2005: стеки, ноды, слайдеры, чекбоксы, меню

среда нативно удобна для CG-художников, но ее сложность быстро растет

2025: сложные интерфейсы с логикой и формулами

сложность проектов и интерфейсов изменила содержание

профессии CG-художника

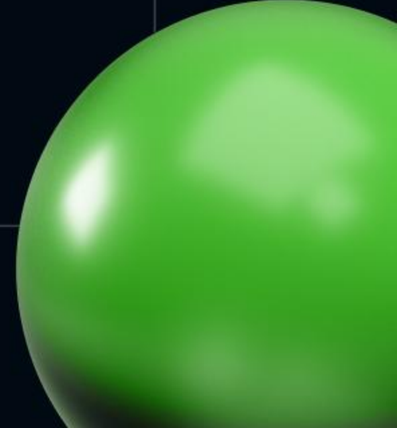
Типичный пайплайн

разработки приложений с большим объемом графики

1. пишется фича

2. делается UI для художников

3. художники подбирают “красивые” параметры



Концепции UI и их проблемы

Inspector (Unity) и Details (Unreal Engine)

+ удобен для редактирования параметров

Концепции UI и их проблемы

Inspector (Unity) и Details (Unreal Engine)

+ удобен для редактирования параметров
– логика работы фиксирована

Концепции UI и их проблемы

Inspector (Unity) и Details (Unreal Engine)

\+ удобен для редактирования параметров

– логика работы фиксирована

? попытки внедрить сложные концепции (связи, формулы и т.д.) плохо ложатся на такой UI

Концепции UI и их проблемы

Ноды (blueprints в Unreal Engine) - попытка вынести в GUI логику и сложное поведение.

\+ красиво выглядит, управляется мышью

Концепции UI и их проблемы

Ноды (blueprints в Unreal Engine) - попытка вынести в GUI логику и сложное поведение.

+ красиво выглядит, управляется мышью
– требует постоянного наведения порядка
– клавиатура все равно требуется
– занимает много места на экране
– сложные схемы нечитаемы

Концепции UI и их проблемы

Ноды (blueprints в Unreal Engine) - попытка вынести в GUI логику и сложное поведение.

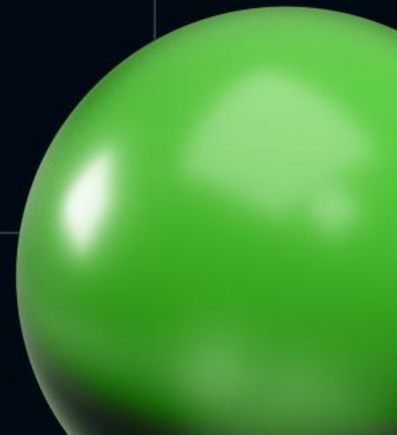
+ красиво выглядит, управляется мышью
– требует постоянного наведения порядка
– клавиатура все равно требуется
– занимает много места на экране
– сложные схемы нечитаемы
? удобство отладки - спорное

Концепции UI и их проблемы

Код

```
pos.x = radius * sin(theta) * cos(phi);
pos.y = radius * cos(theta);
pos.z = radius * sin(theta) * sin(phi);
```

+ Концепции любой сложности
+ Компактность
+ Произвольное форматирование
+ Развитые средства отладки



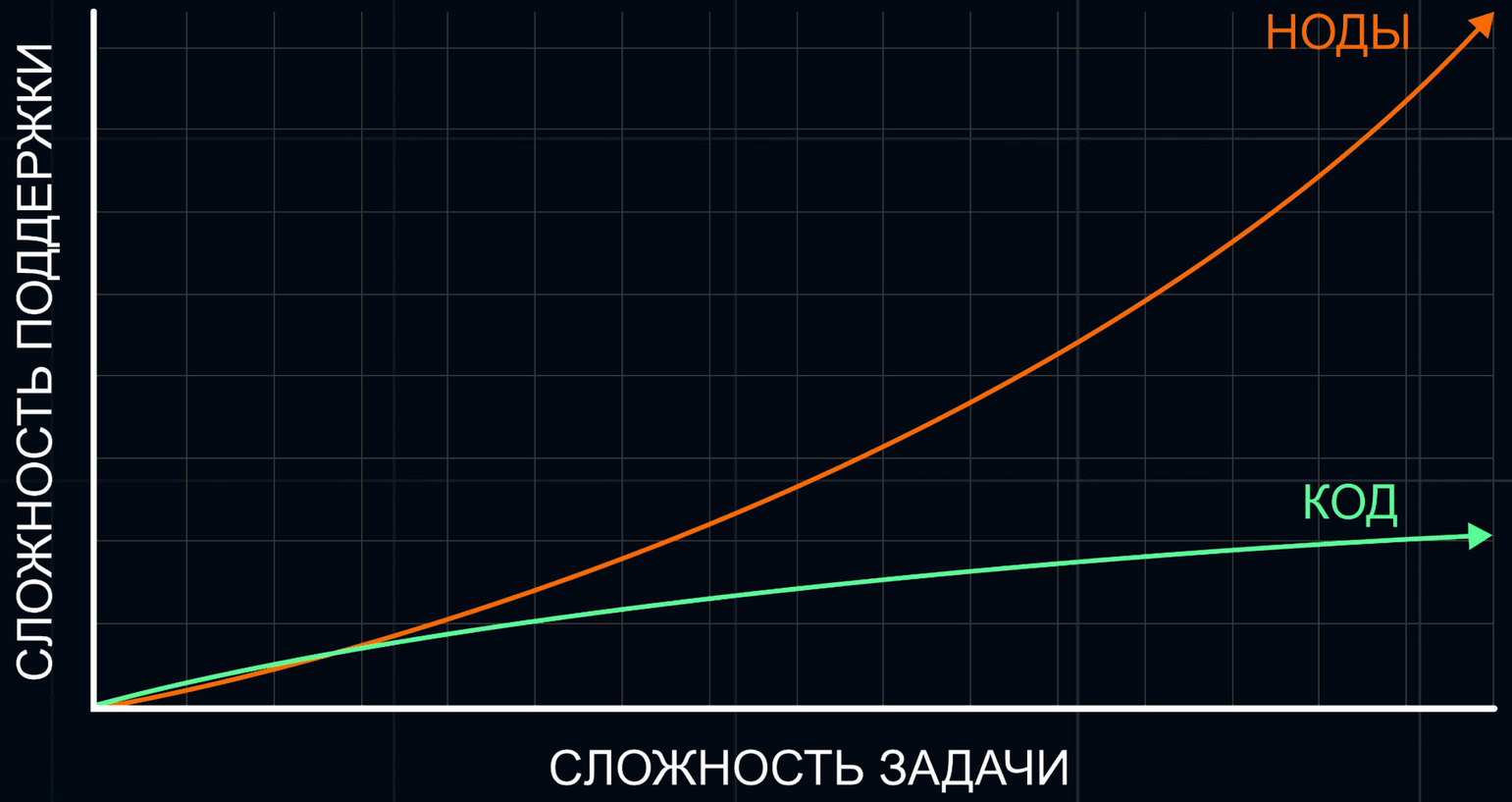
Концепции UI и их проблемы

Код

```
pos.x = radius * sin(theta) * cos(phi);
pos.y = radius * cos(theta);
pos.z = radius * sin(theta) * sin(phi);
```

+ Концепции любой сложности
+ Компактность
+ Произвольное форматирование
+ Развитые средства отладки
– Нет возможности менять параметры мышью
– Нет возможности менять параметры в реальном времени

Концепции UI и их проблемы

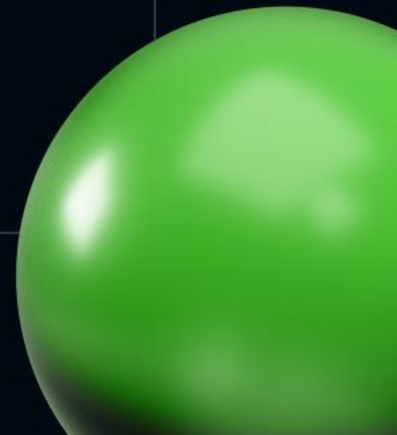


Концепции UI и их проблемы

Решение - апгрейд текстового UI Visual Studio:

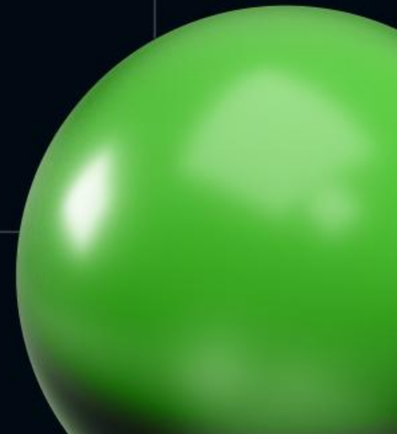
+ возможность изменять параметры мышкой
+ возможность изменять их в реальном времени

Остальное есть в IDE “из коробки”.



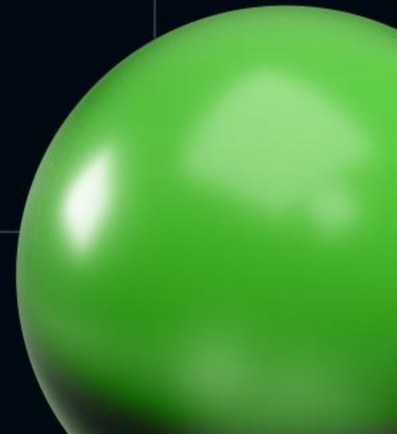
Требуемая скорость реакции

	Изменение конвейера (c++)	Добавление параметров для шейдера	Изменение кода шейдера	Изменение параметров отрисовки (c++)
частота изменений	редко	периодически	часто	очень часто
требуемое время реакции	не критично	не критично	не более секунды	не более 16мс



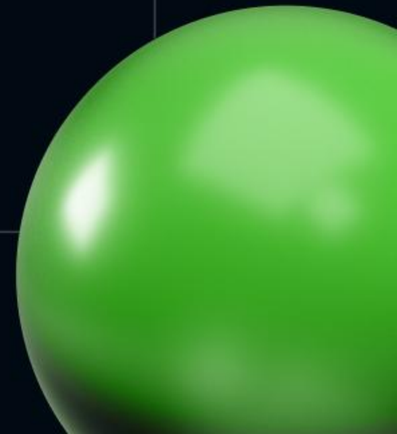
Требуемая скорость реакции

	Изменение конвейера (c++)	Добавление параметров для шейдера	Изменение кода шейдера	Изменение параметров отрисовки (c++)
частота изменений	редко	периодически	часто	очень часто
требуемое время реакции	не критично	не критично	не более секунды	не более 16мс



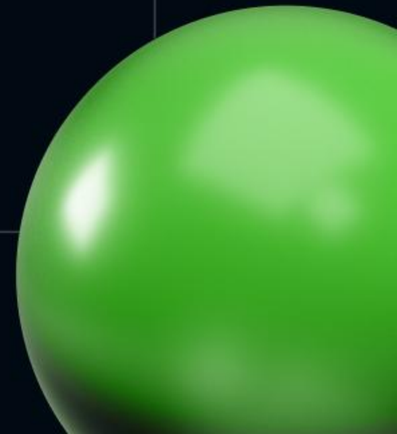
Требуемая скорость реакции

	Изменение конвейера (c++)	Добавление параметров для шейдера	Изменение кода шейдера	Изменение параметров отрисовки (c++)
частота изменений	редко	периодически	часто	очень часто
требуемое время реакции	не критично	не критично	не более секунды	не более 16мс

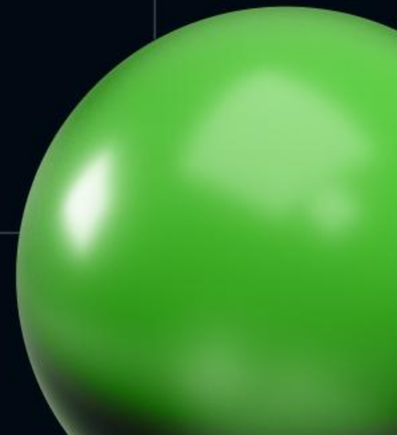


Требуемая скорость реакции

	Изменение конвейера (c++)	Добавление параметров для шейдера	Изменение кода шейдера	Изменение параметров отрисовки (c++)
частота изменений	редко	периодически	часто	очень часто
требуемое время реакции	не критично	не критично	не более секунды	не более 16мс



Livecoding-сессия



Архитектура решения

шейдеры, редактирование средствами VS

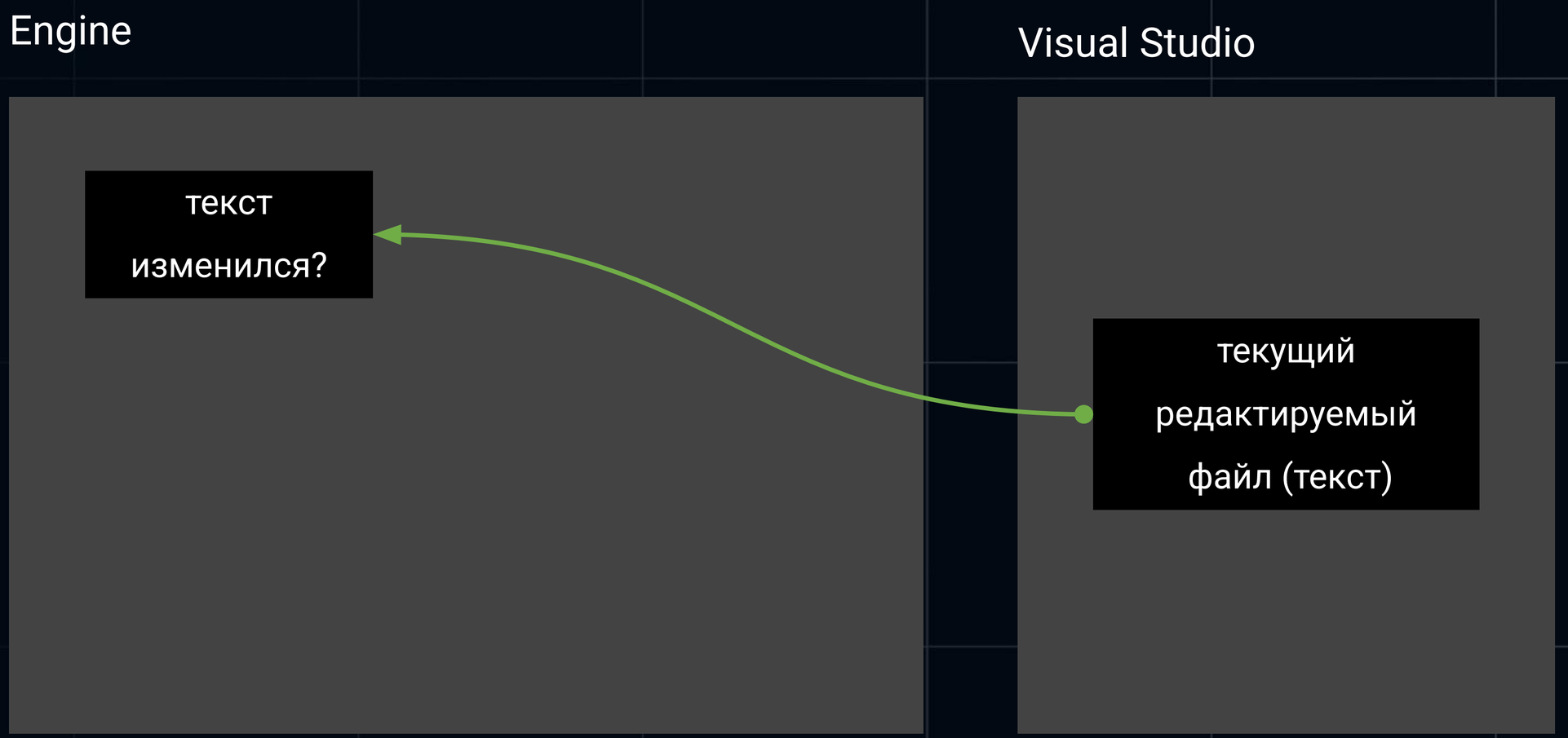
Engine

Visual Studio

текущий редактируемый файл (текст)

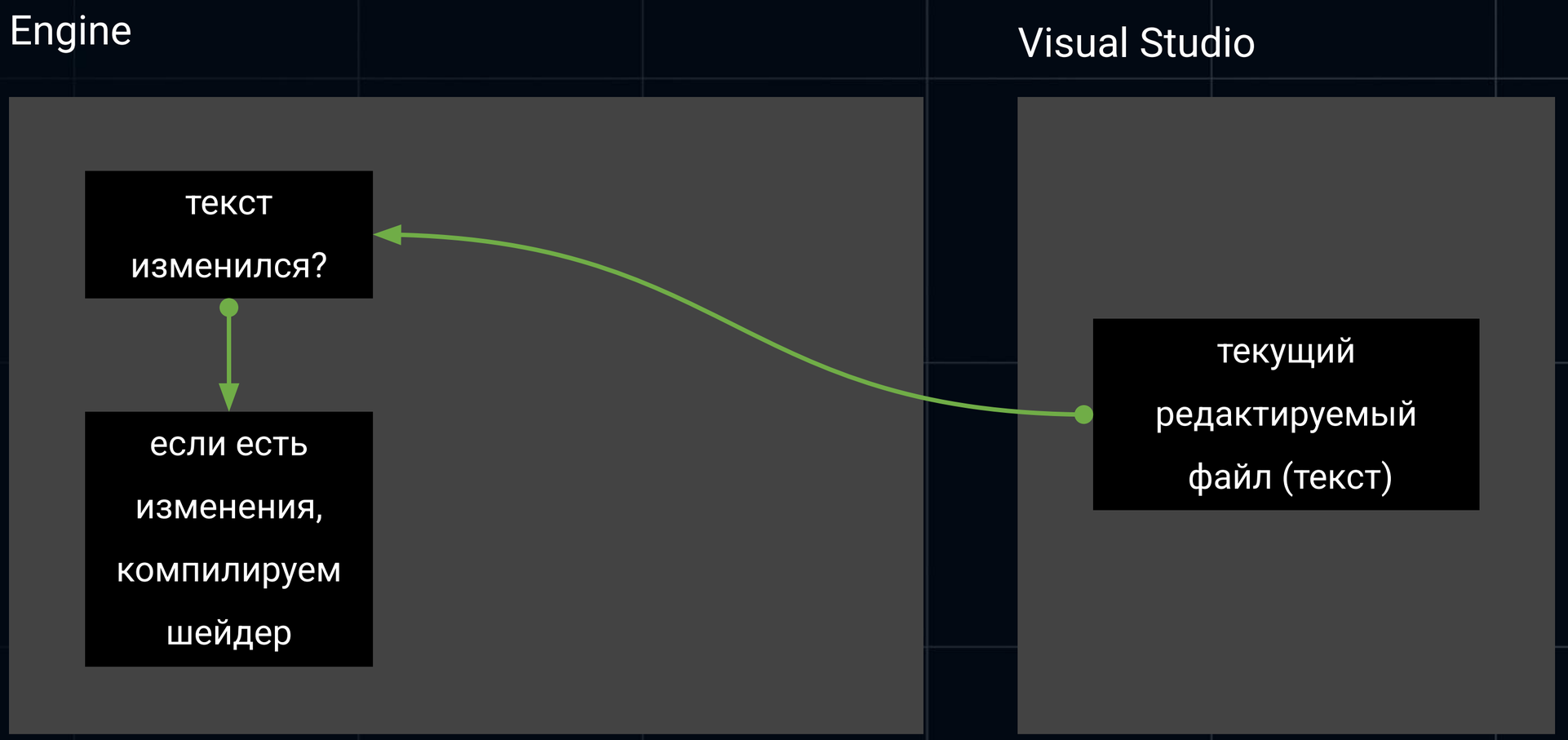
Архитектура решения

шейдеры, редактирование средствами VS



Архитектура решения

шейдеры, редактирование средствами VS



Архитектура решения

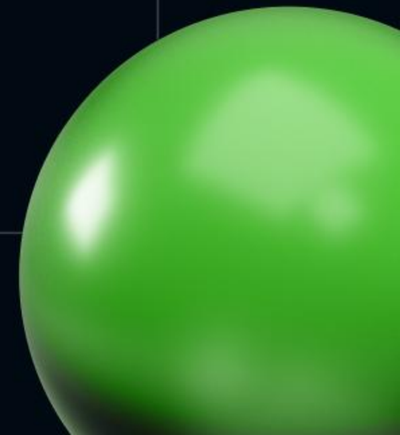
шейдеры, изменение параметров мышью

Engine

Visual Studio

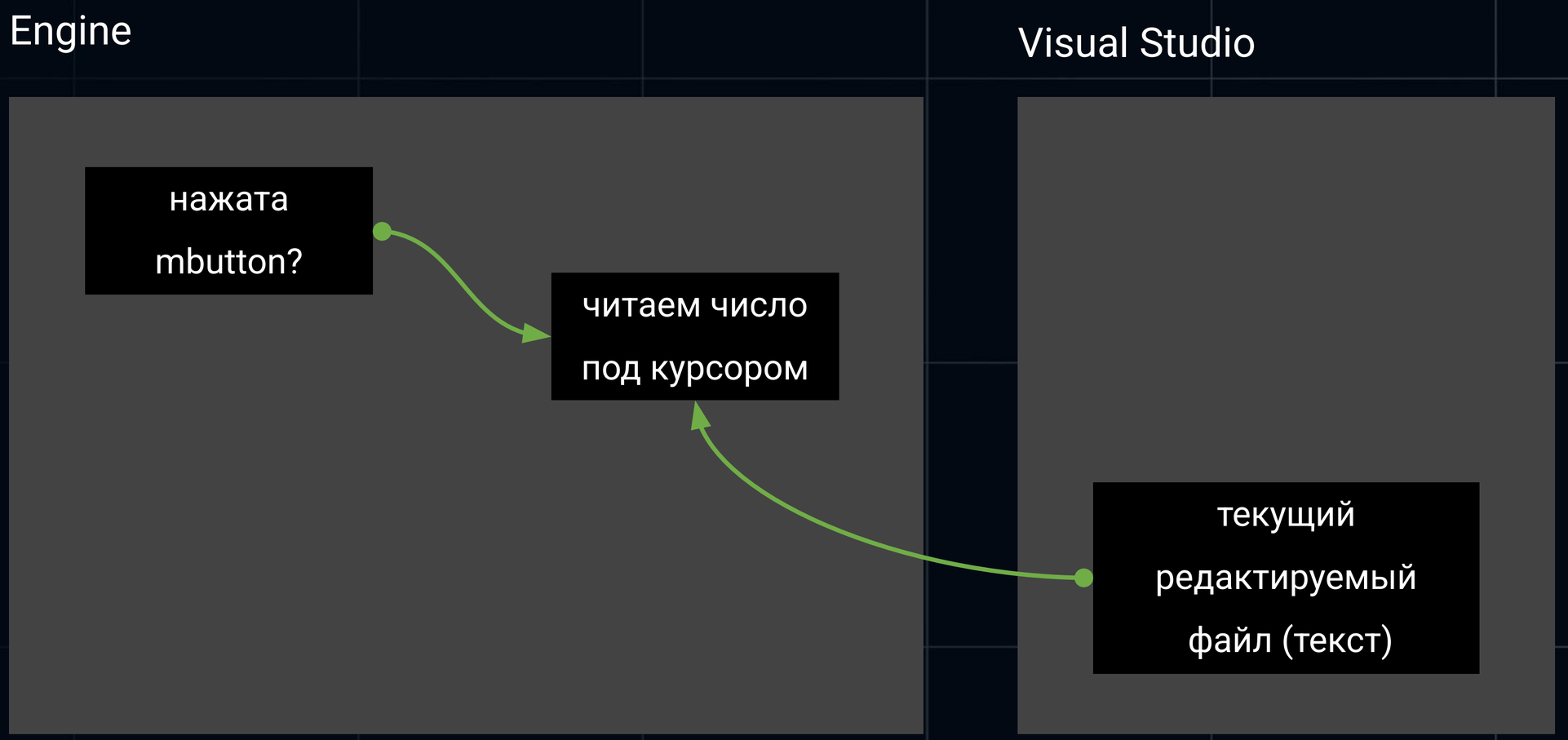
нажата
mbutton?

текущий
редактируемый
файл (текст)



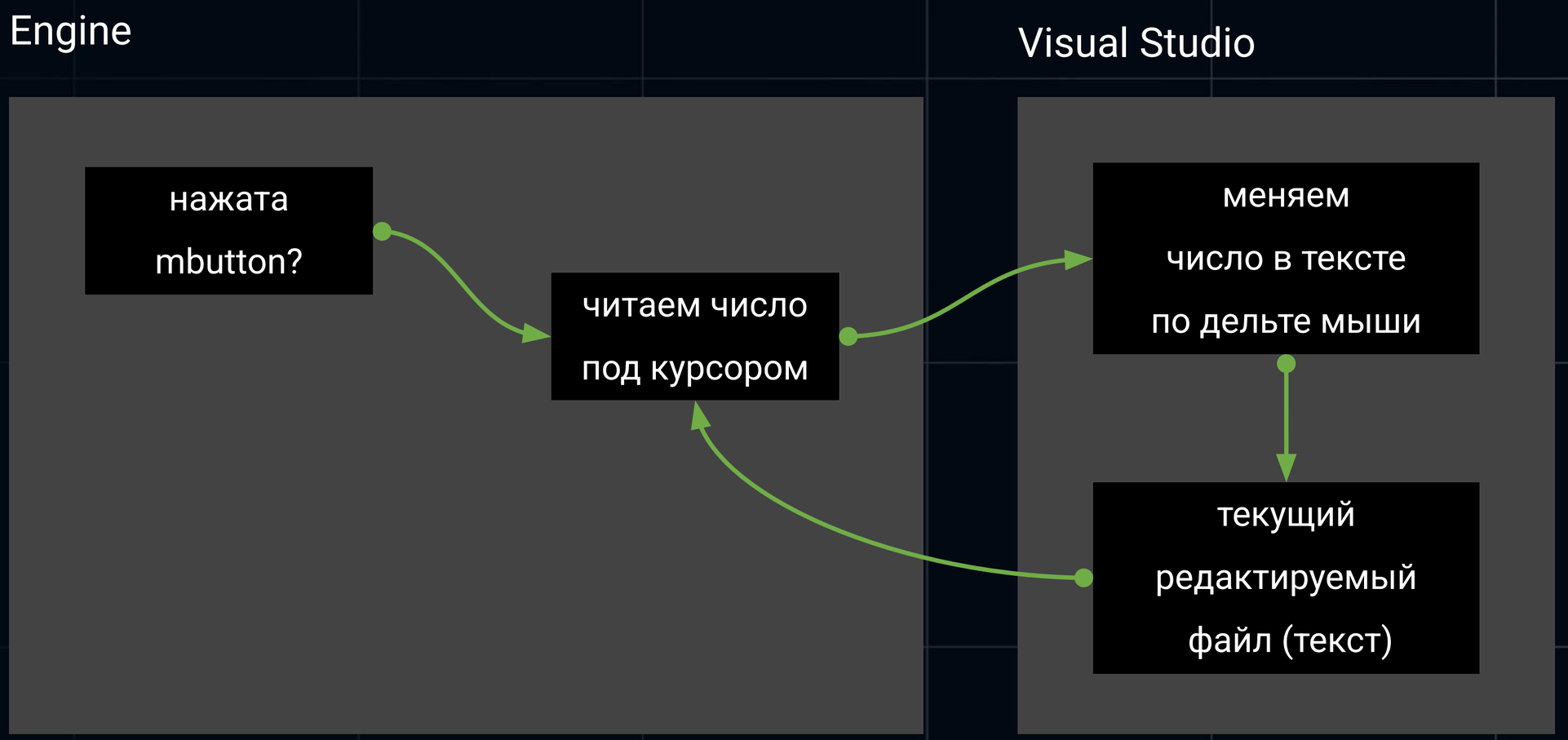
Архитектура решения

шейдеры, изменение параметров мышью



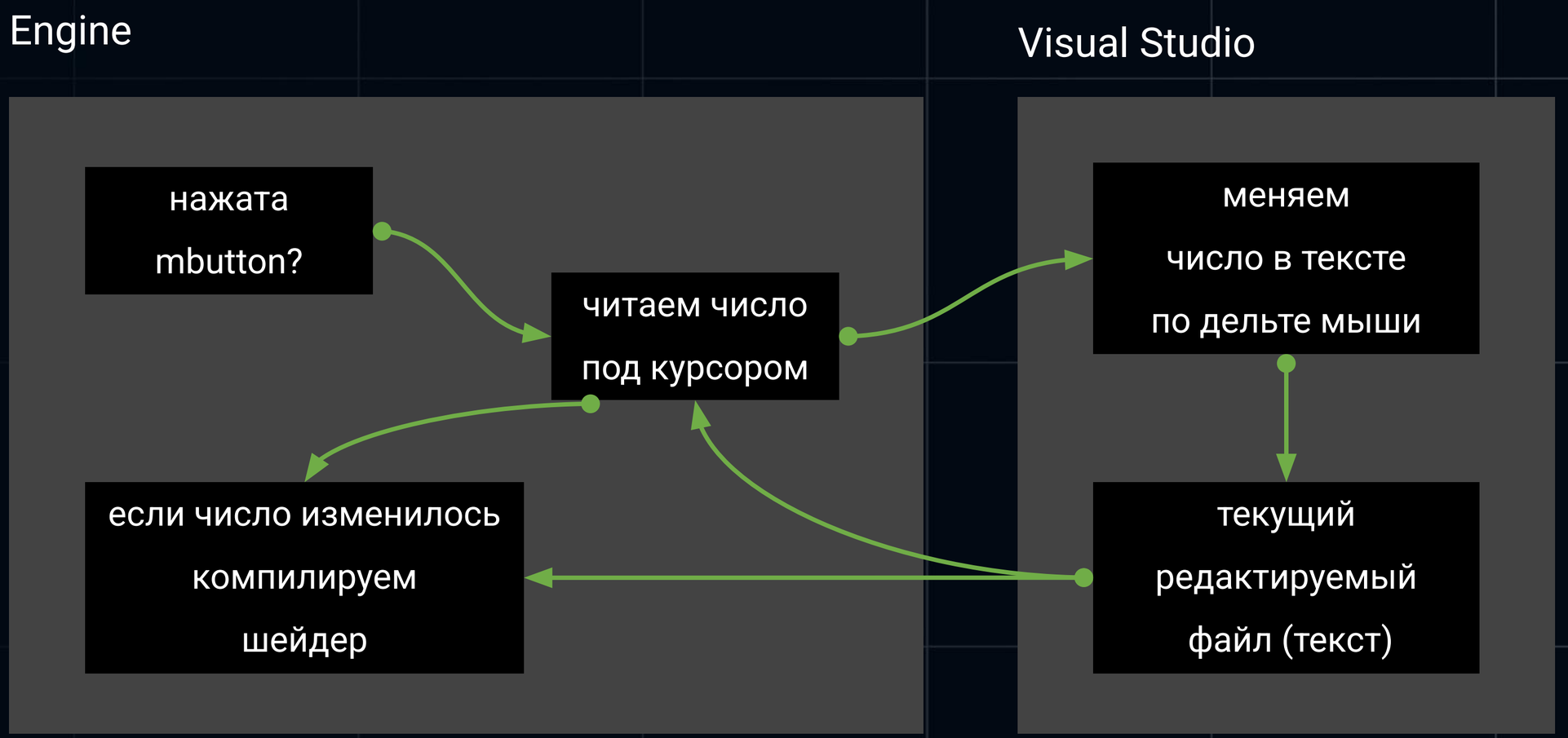
Архитектура решения

шейдеры, изменение параметров мышью



Архитектура решения

шейдеры, изменение параметров мышью



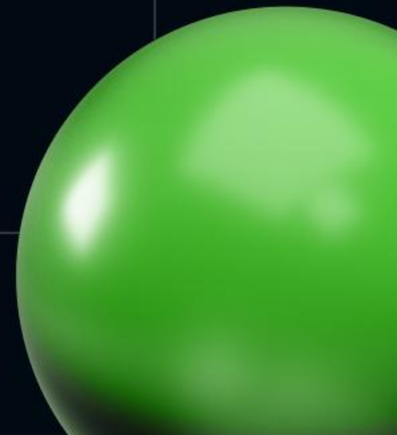
Интеграция шейдеров

- Передача параметров - принцип SSoT

  Константные буферы должны декларироваться в одном месте.

  По pre-build event генерируем c++ структуры исходя из текста констант-буфера.

  * Почему не подходит стандартная рефлексия DirectX?



Интеграция шейдеров

- Передача параметров - принцип SSoT

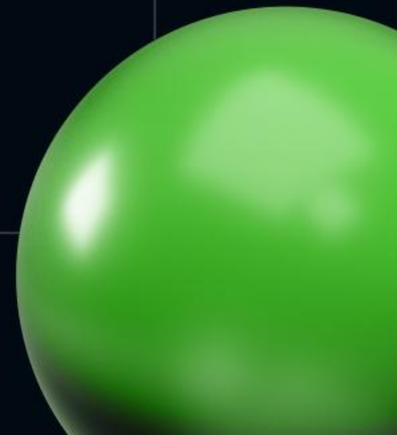
  Константные буферы должны декларироваться в одном месте.

  По pre-build event генерируем c++ структуры исходя из текста констант-буфера.

  * Почему не подходит стандартная рефлексия DirectX?

- Фоновая компиляция

  Синхронизация через файлы (автосохранение + вотчер) или трекинг изменений текущего редактируемого в Visual Studio файла (COM/EnvDTE).



Интеграция шейдеров

- Передача параметров - принцип SSoT

  Константные буферы должны декларироваться в одном месте.

  По pre-build event генерируем c++ структуры исходя из текста констант-буфера.

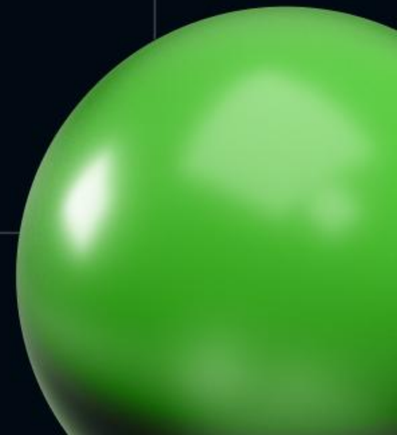
  * Почему не подходит стандартная рефлексия DirectX?

- Фоновая компиляция

  Синхронизация через файлы (автосохранение + вотчер) или трекинг изменений текущего редактируемого в Visual Studio файла (COM/EnvDTE).

- Удобный доступ к списку ресурсов

  По pre-build event генерируем enum классы для текстур, шейдеров, комплексных объектов. Автокомплит позволяет видеть имя при выборе.



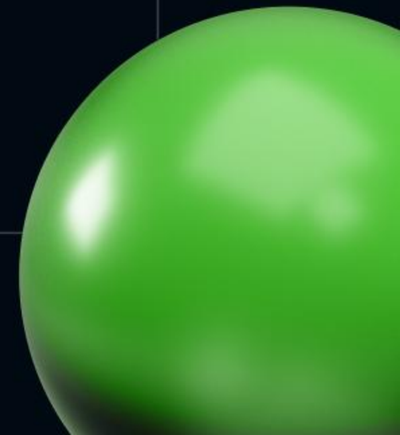
Архитектура решения

c++, рефлексивная структура

Как менять параметры вызова в рантайме без перекомпиляции?

Допустим, мы вызываем функцию:

```
func(1, 2);
```



Архитектура решения

c++, рефлексивная структура

Как менять параметры вызова в рантайме без перекомпиляции?

Допустим, мы вызываем функцию:

```
func(1, 2);
```

- Поскольку нет доступа к передаваемым значениям средствами языка, мы должны обеспечить их подмену - через чтение из рефлексивной структуры.

Архитектура решения

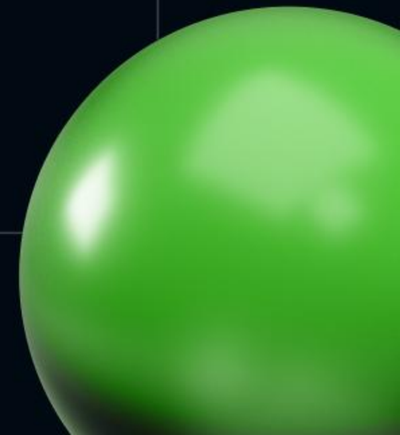
c++, рефлексивная структура

Как менять параметры вызова в рантайме без перекомпиляции?

Допустим, мы вызываем функцию:

```
func(1, 2);
```

- Поскольку нет доступа к передаваемым значениям средствами языка, мы должны обеспечить их подмену - через чтение из рефлексивной структуры.
- рефлексивная структура инициализируется на старте



Архитектура решения

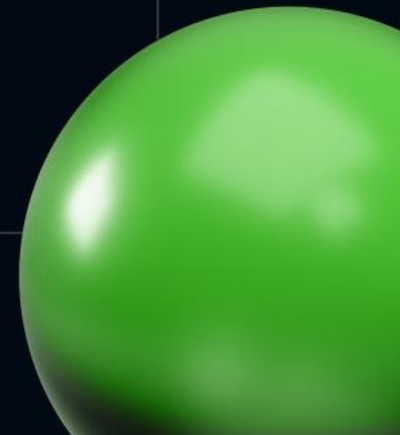
c++, рефлексивная структура

Как менять параметры вызова в рантайме без перекомпиляции?

Допустим, мы вызываем функцию:

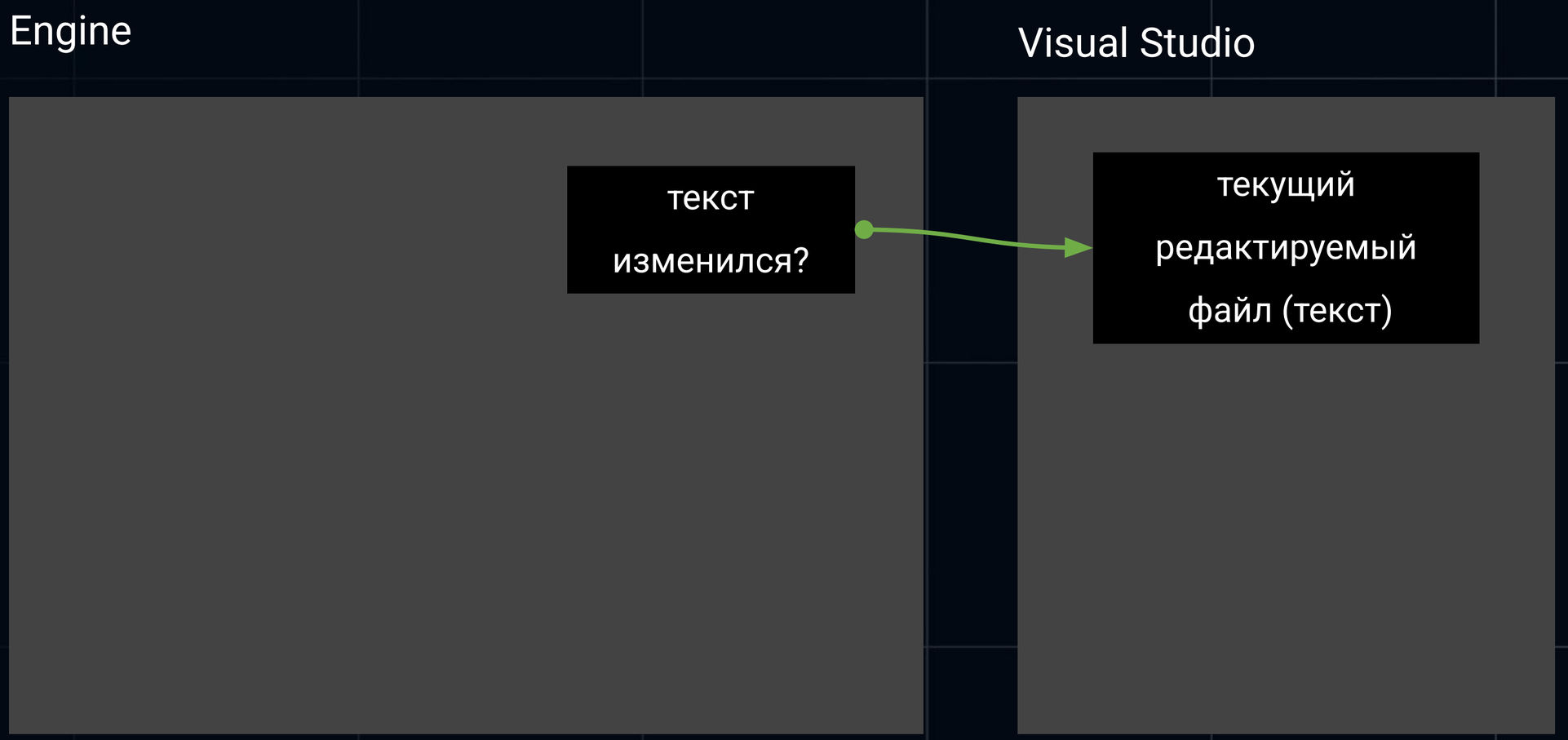
```
func(1, 2);
```

- Поскольку нет доступа к передаваемым значениям средствами языка, мы должны обеспечить их подмену - через чтение из рефлексивной структуры.
- рефлексивная структура инициализируется на старте
- текст программы и рефлексивная структура всегда должны быть синхронизированы



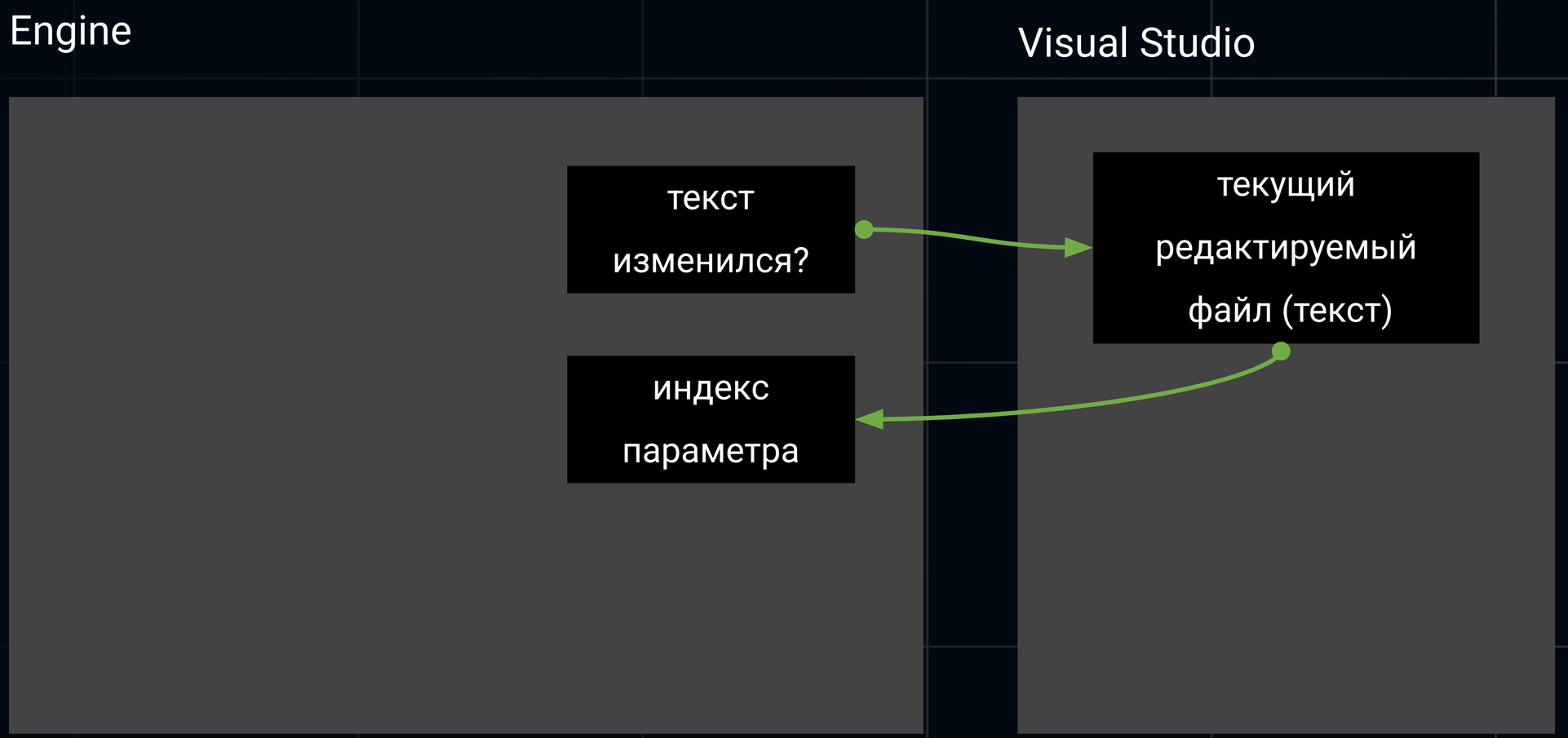
Архитектура решения

c++, редактирование средствами VS



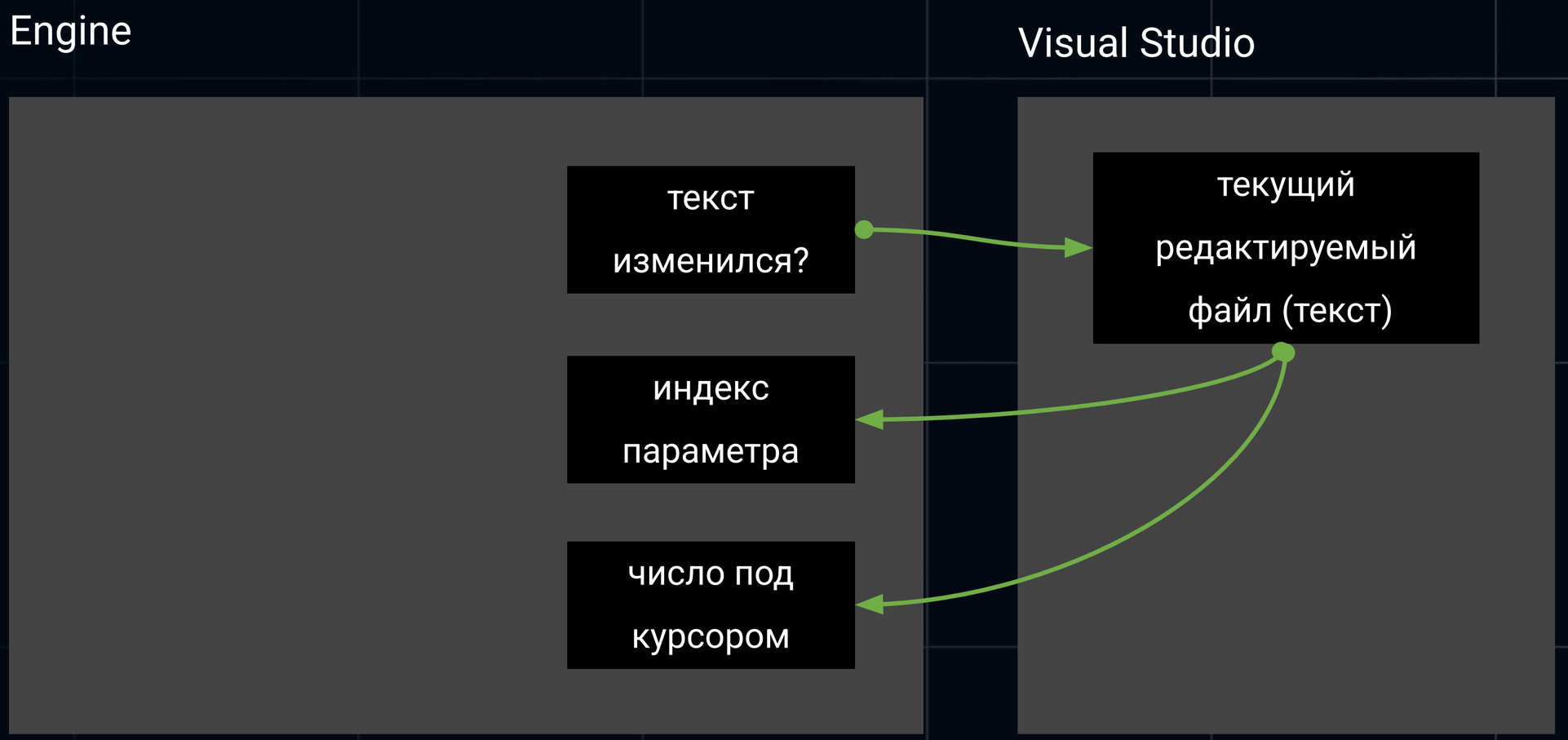
Архитектура решения

c++, редактирование средствами VS



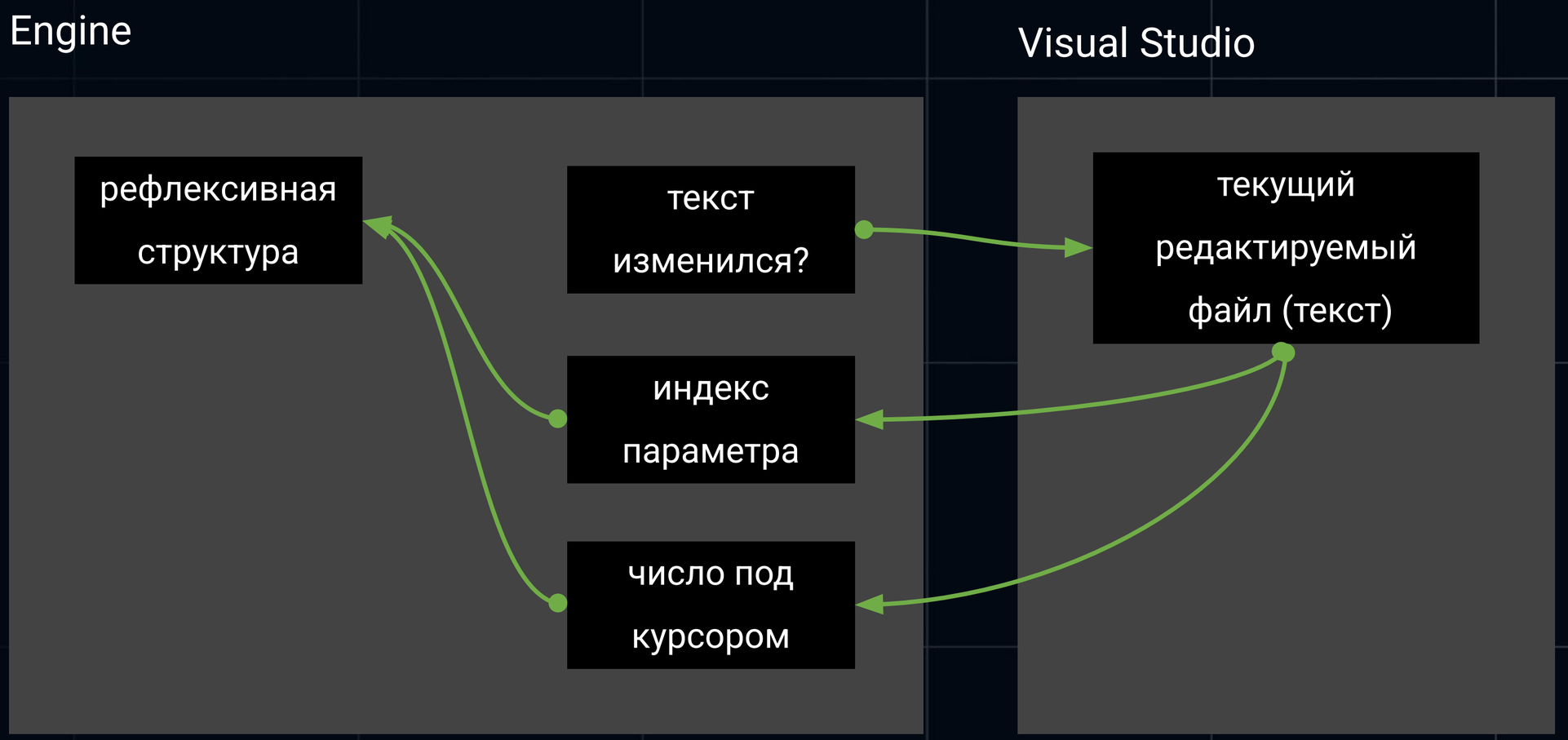
Архитектура решения

c++, редактирование средствами VS



Архитектура решения

c++, редактирование средствами VS



Архитектура решения

c++, изменение параметров мышью

Engine

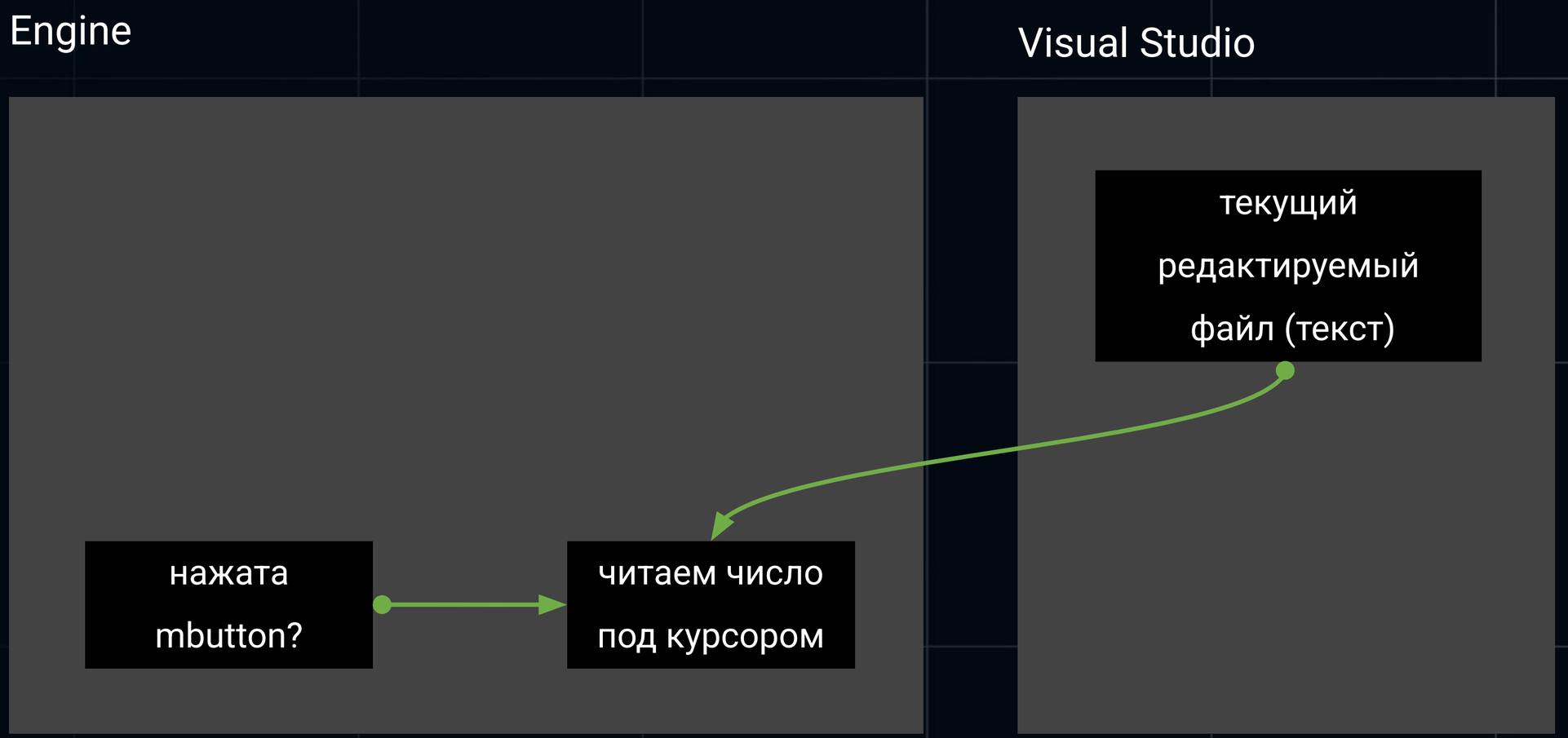
нажата
mbutton?

Visual Studio

текущий
редактируемый
файл (текст)

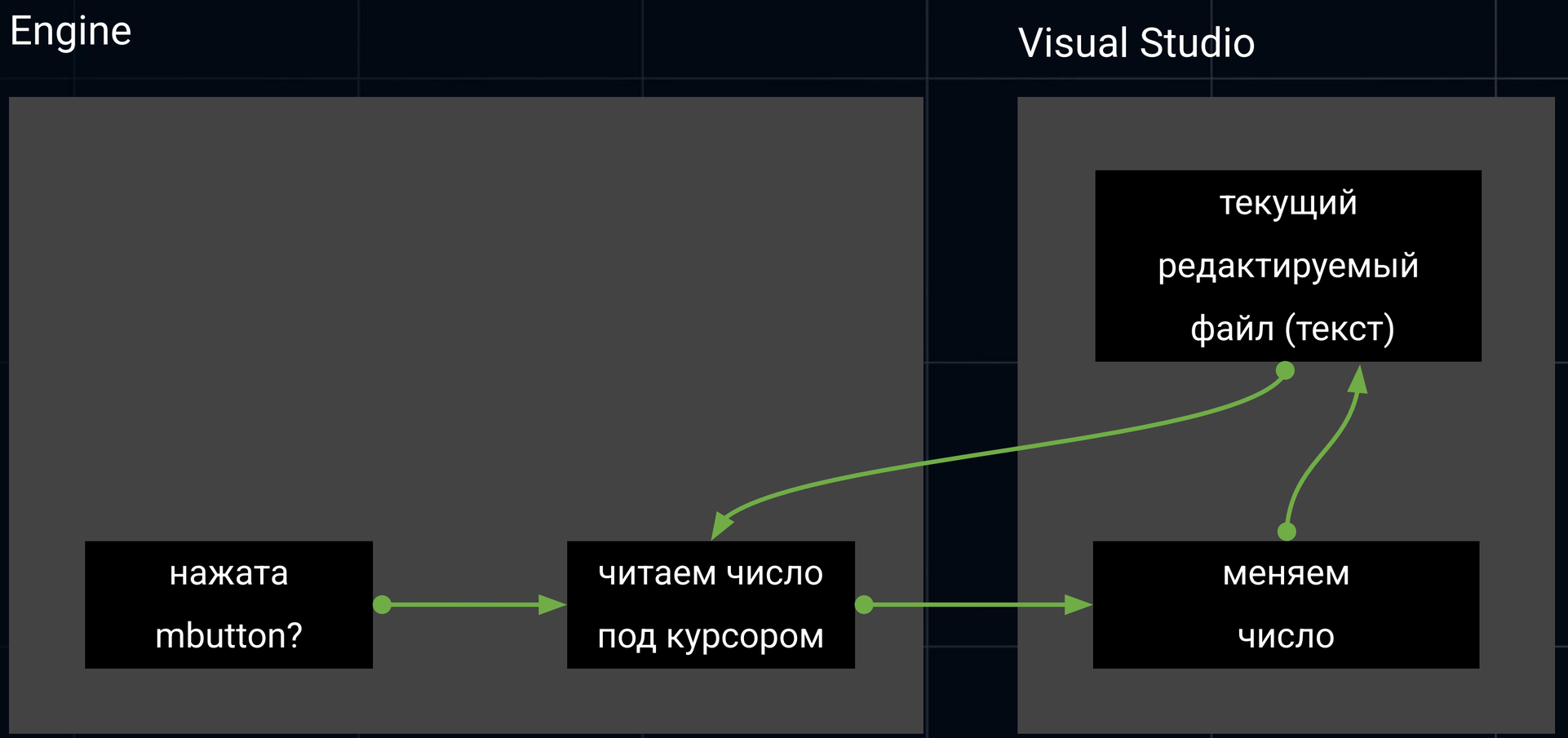
Архитектура решения

c++, изменение параметров мышью



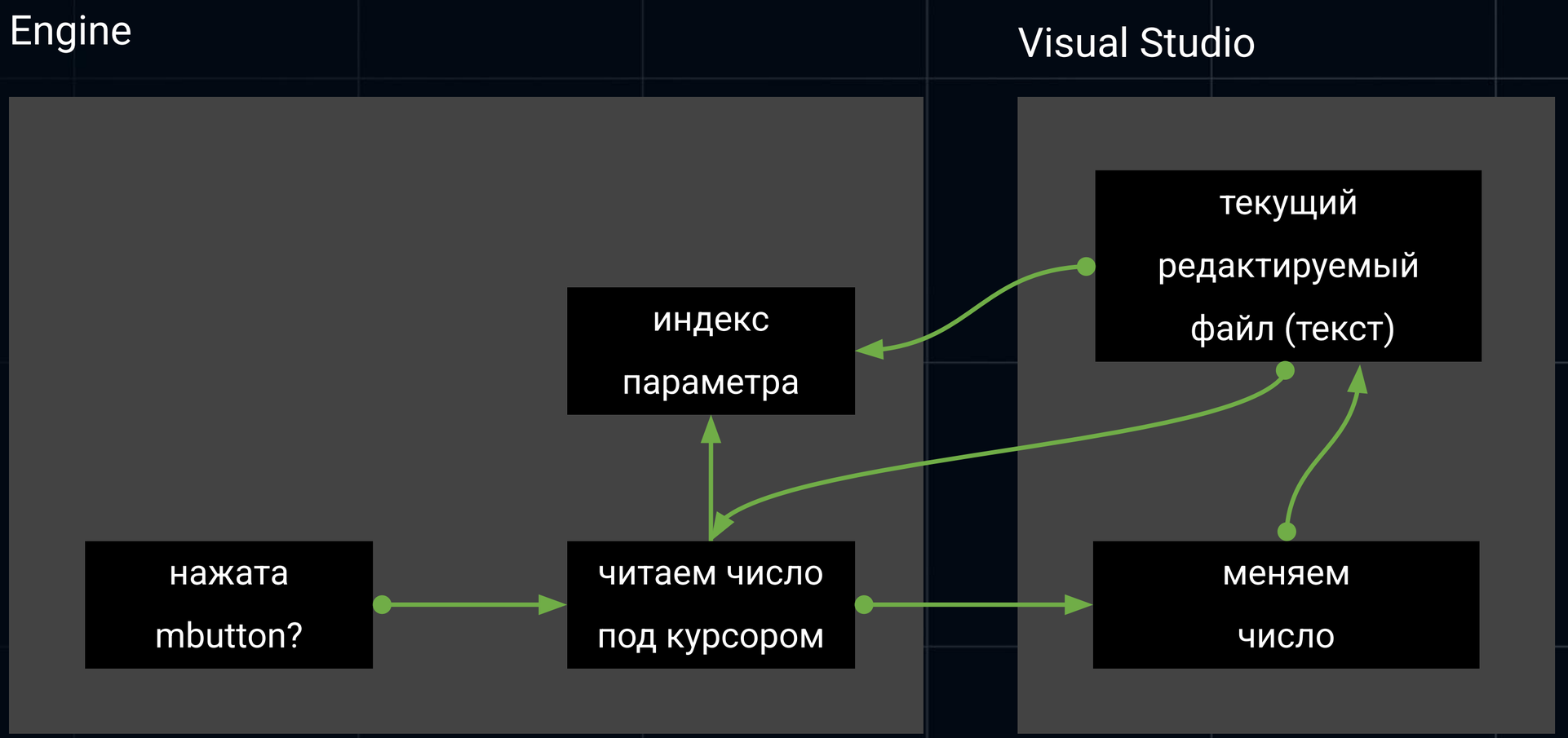
Архитектура решения

c++, изменение параметров мышью



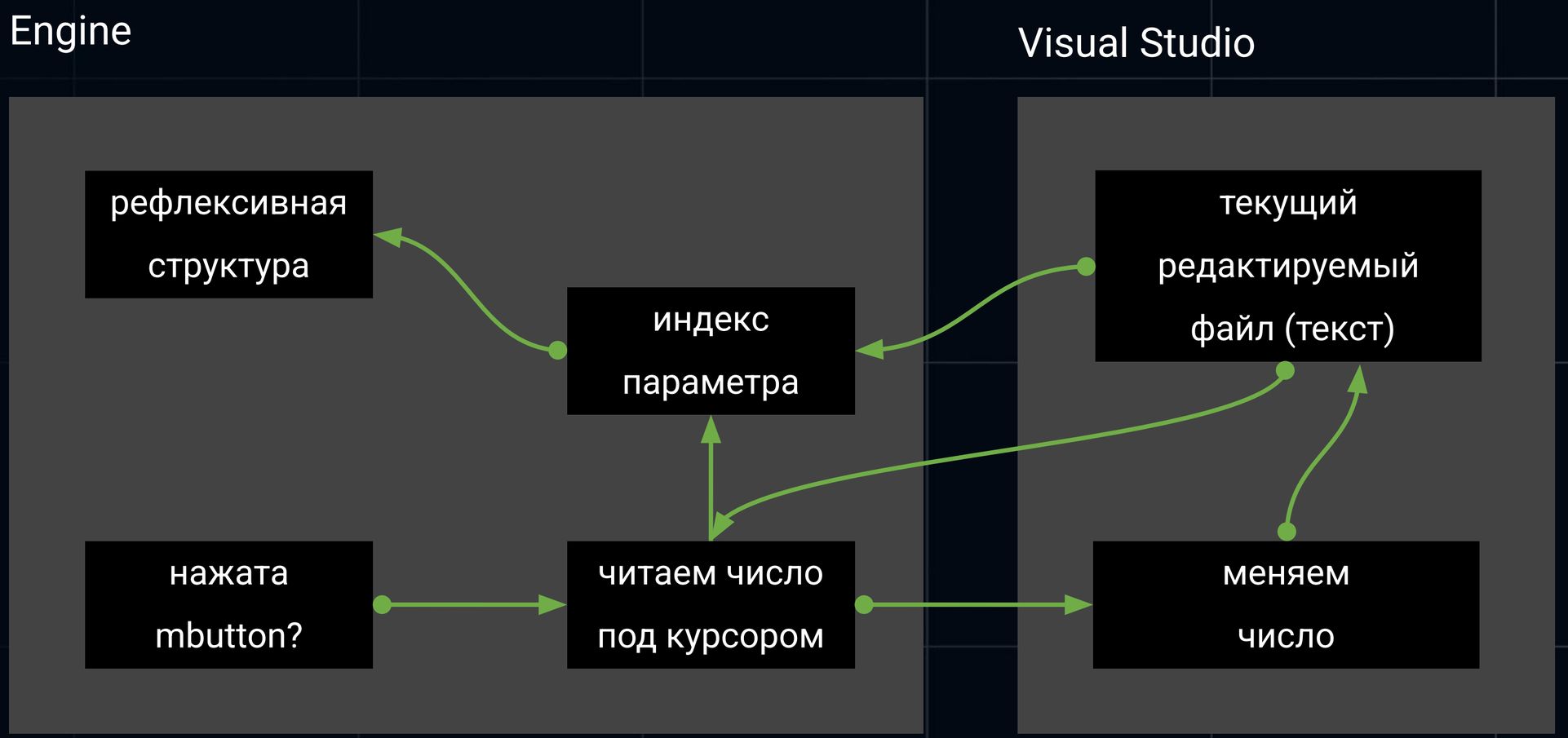
Архитектура решения

c++, изменение параметров мышью



Архитектура решения

c++, изменение параметров мышью

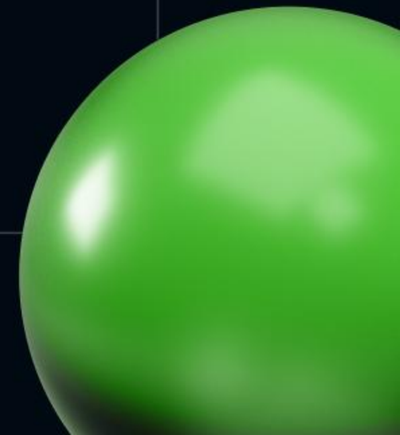


Рефлексия в C++

Синтаксические соглашения. Агрегатная инициализация.

декларация:

```
namespace Object {
    cmd(Girl, int quality,
          int xPos, int yPos, int zPos)
    {
        reflect; //вызов макроса
        …
        //тело функции
    }
}
```



Рефлексия в C++

Синтаксические соглашения. Агрегатная инициализация.

декларация:

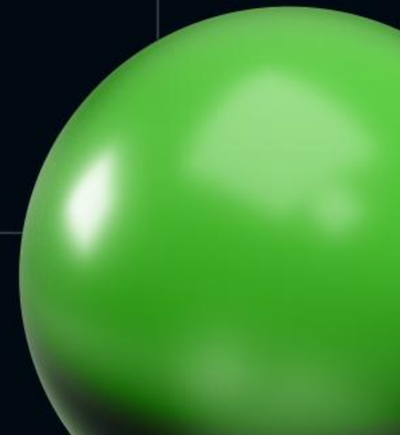
```
namespace Object {
    cmd(Girl, int quality,
          int xPos, int yPos, int zPos)
    {
        reflect; //вызов макроса
        …
        //тело функции
    }
}
```

вызов:

```
Object::Girl({
        .quality = 1,
        .xPos = -25,
        .yPos = 165,
        .zPos = 0,
});
```

Скобки позволяют скрыть параметры

с помощью +/- в IDE



Рефлексия в C++

Развертывание макросов

```
#define cmd(name, ...) struct CAT(name,_params) { FOR_EACH(SEMI, __VA_ARGS__) }; \
    void name(CAT(name,_params) in)
```

декларация cmd(func, int a, int b) развернется в

Snippet

```
struct func { int a; int b; };
void func(func_params in, const std::source_location caller = std::source_location::current())
```

- Теперь функция знает, откуда ее вызвали (файл, строка) и может выполнить рефлексию, прочитав свой исходный код (список параметров и их типов) и код вызова (сами параметры)

Рефлексия в C++

Развертывание макросов - загрузка или подмена параметров

```
void reflect_f(auto* in, const std::source_location caller, const std::source_location currentFunc);
#define reflect editor::paramEdit::reflect_f(&in, caller, std::source_location::current())
```

внутри функции reflect_f:

Snippet

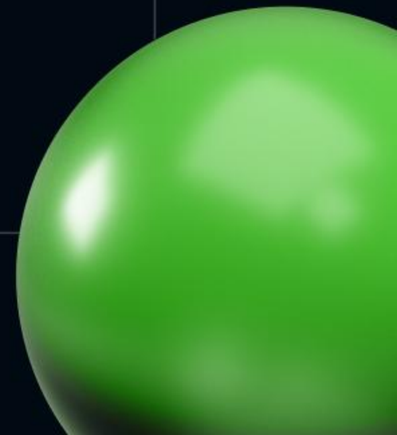
```
    if (loaded) {
        //поля структуры нашей функции ← рефлексивная структура
    } else {
        //исходный текст → рефлексивная структура
    }
```

(!) для вычисляемых значений ставится флаг bypass, и они никак не изменяются

Рефлексия в C++

Реестр для корректной работы условных вызовов

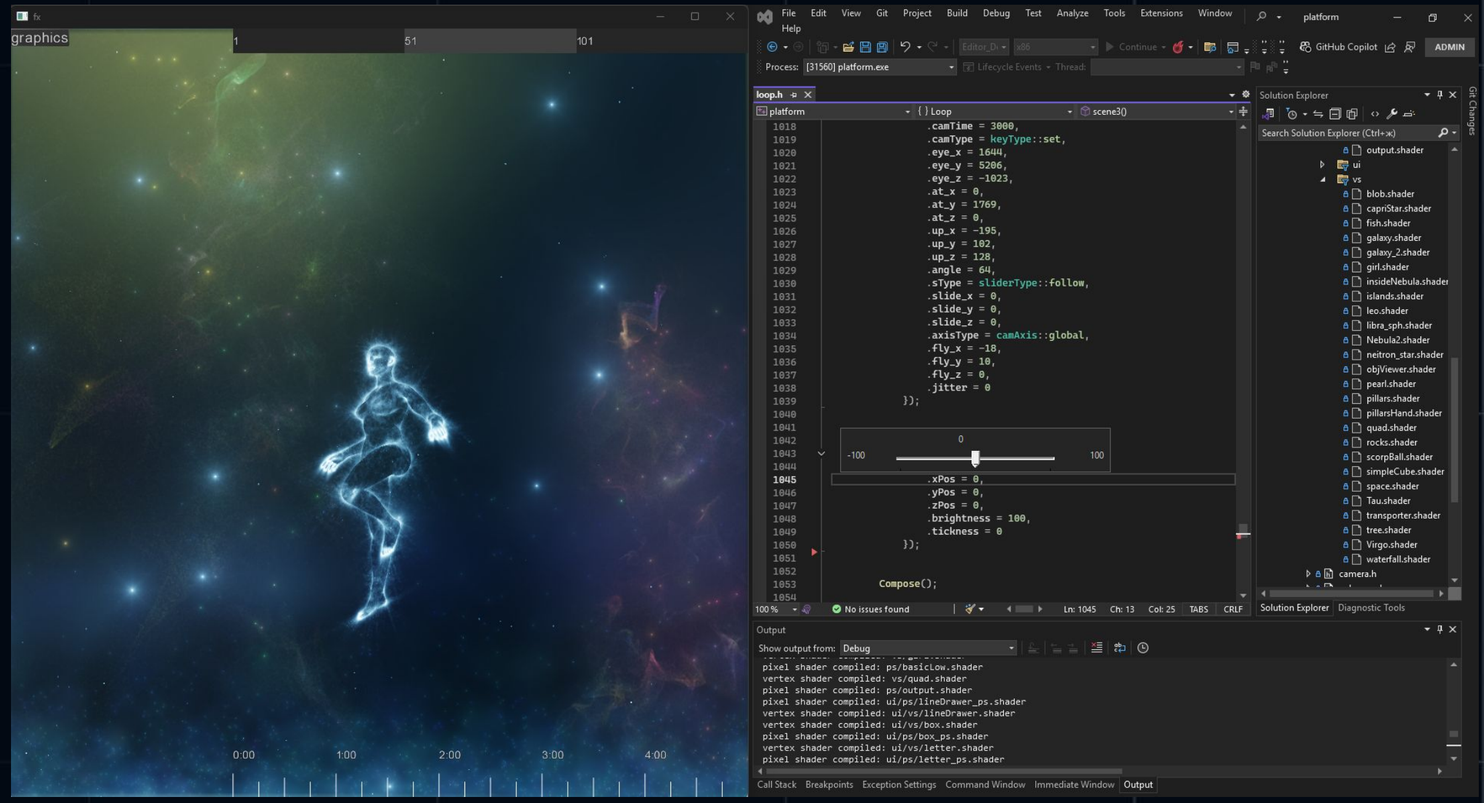
```
static std::unordered_map<std::string, int> registry;
void reflect_f(auto* in, const std::source_location caller, const std::source_location currentFunc) {
    std::string key = std::string(caller.file_name()) + ":" + std::to_string(caller.line());
    auto it = registry.find(key);
    if (it != registry.end()) {
        cmdCounter = registry[key];
        cmdParamDesc[cmdCounter].loaded = true;
    } else {
        cmdCounter = registry[key] = next_index++;
        cmdParamDesc[cmdCounter].loaded = false;
    }
}
```



UI:Окна

Движок автоматически располагает окно просмотра слева, а окно Visual Studio справа.

После завершения работы окно восстанавливает исходное положение

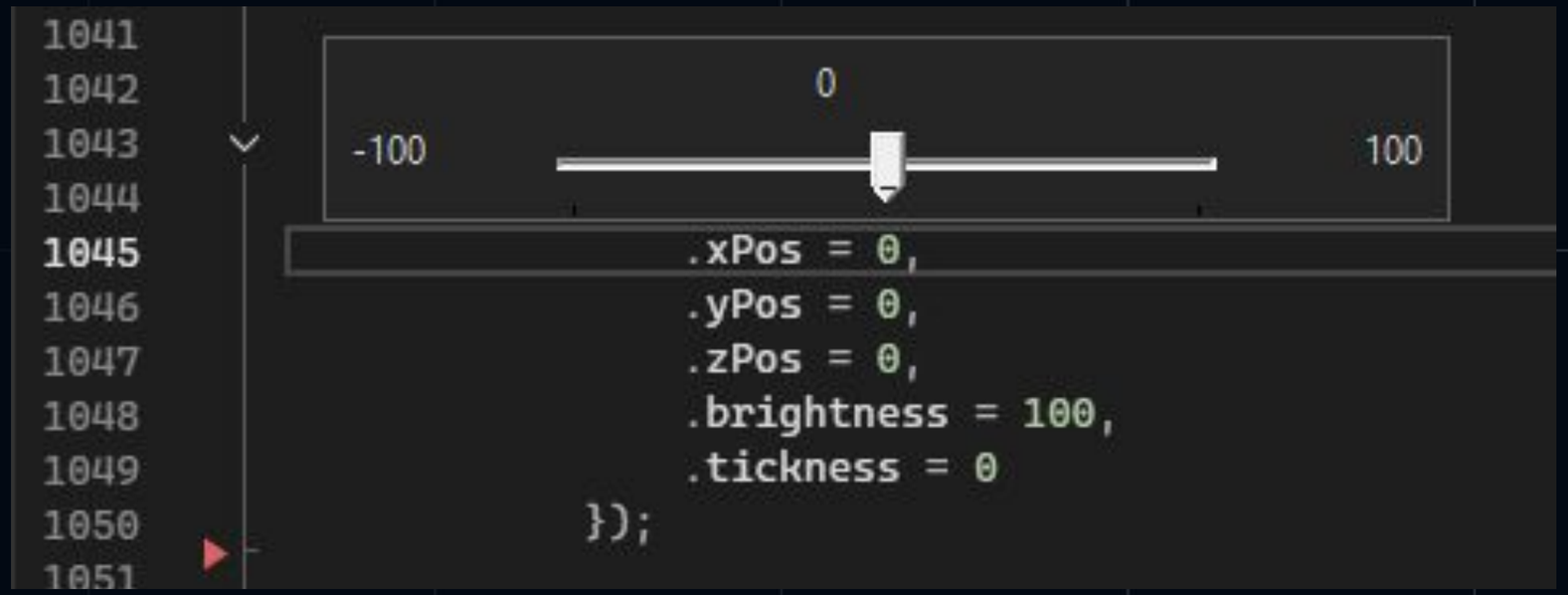


* Для обладателей двухмониторных конфигураций - окно просмотра запускается на втором экране

UI:Слайдер

Вызываем и убираем слайдер по нажатию средней кнопки мышки, рисуем его в текущей позиции курсора. Альтернативный способ - click&drag средней кнопкой мышки.

Прочие варианты конфликтуют с UI Visual Studio и/или требуют глобальных хуков, которые плохо уживаются с точками останова и эксепшнами.



Работа с VS (COM/EnvDTE)

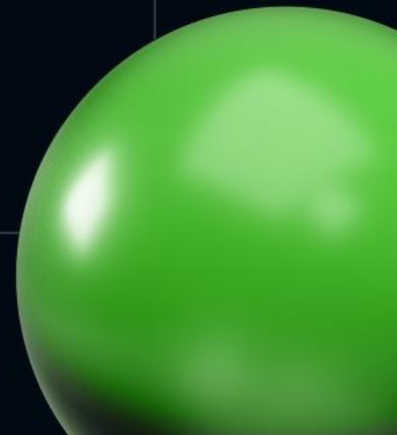
Возможности взаимодействия с Visual Studio

- Получение имени текущего редактируемого файла

Работа с VS (COM/EnvDTE)

Возможности взаимодействия с Visual Studio

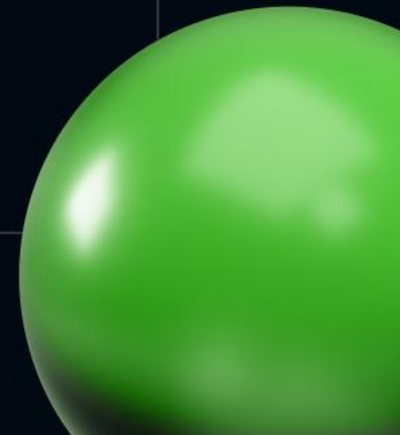
- Получение имени текущего редактируемого файла
- Получение позиции курсора



Работа с VS (COM/EnvDTE)

Возможности взаимодействия с Visual Studio

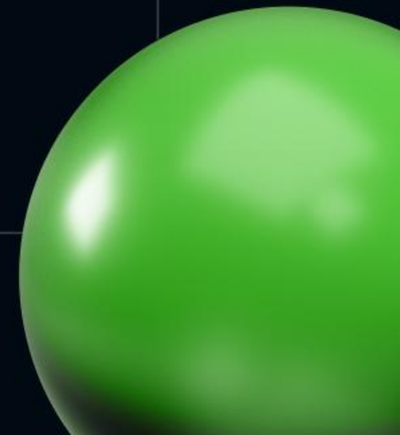
- Получение имени текущего редактируемого файла
- Получение позиции курсора
- Получение текста всего файла или его части



Работа с VS (COM/EnvDTE)

Возможности взаимодействия с Visual Studio

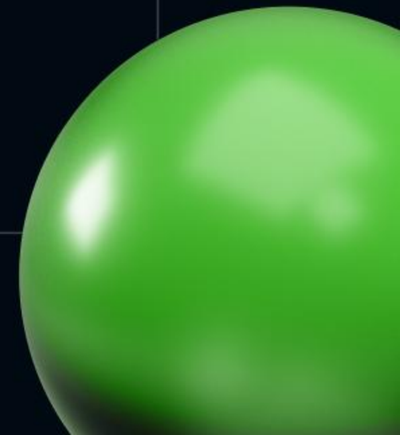
- Получение имени текущего редактируемого файла
- Получение позиции курсора
- Получение текста всего файла или его части
- Получение статуса несохраненных изменений



Работа с VS (COM/EnvDTE)

Возможности взаимодействия с Visual Studio

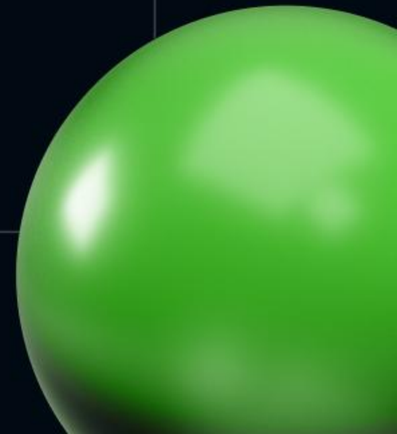
- Получение имени текущего редактируемого файла
- Получение позиции курсора
- Получение текста всего файла или его части
- Получение статуса несохраненных изменений
- Замена текста или его части



Работа с VS (COM/EnvDTE)

Возможности взаимодействия с Visual Studio

- Получение имени текущего редактируемого файла
- Получение позиции курсора
- Получение текста всего файла или его части
- Получение статуса несохраненных изменений
- Замена текста или его части
- ...



Работа с VS (COM/EnvDTE)

Пример. Получение имени текущего редактируемого файла

```
CLSID clsid;
::CLSIDFromProgID(L"VisualStudio.DTE", &clsid);
CComPtr<IUnknown> punk;
::GetActiveObject(clsid, NULL, &punk);
CComPtr<EnvDTE::_DTE> DTE = punk;
CComPtr<EnvDTE::ItemOperations> item_ops;
DTE->get_ItemOperations(&item_ops);
CComPtr<EnvDTE::Document> doc;
DTE->get_ActiveDocument(&doc);
CComBSTR _fileName;
doc->get_FullName(&_fileName);
```

Работа с VS (COM/EnvDTE)

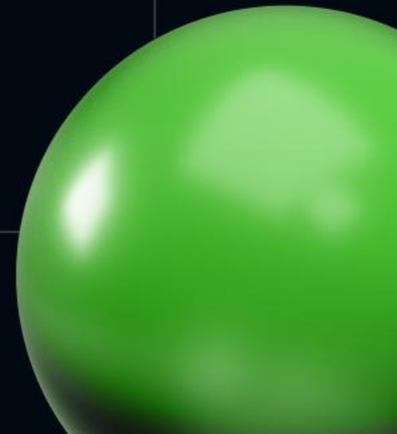
Пример. Получение имени текущего редактируемого файла

```
CLSID clsid;
::CLSIDFromProgID(L"VisualStudio.DTE", &clsid);
CComPtr<IUnknown> punk;
::GetActiveObject(clsid, NULL, &punk);
CComPtr<EnvDTE::_DTE> DTE = punk;
CComPtr<EnvDTE::ItemOperations> item_ops;
DTE->get_ItemOperations(&item_ops);
CComPtr<EnvDTE::Document> doc;
DTE->get_ActiveDocument(&doc);
CComBSTR _fileName;
doc->get_FullName(&_fileName);
```

(!) Чтобы избежать конфликтов
и зависимостей в библиотеках,
а также обеспечить работу
на любой версии студии, лучше
использовать динамический вызов
через IDispatch (позднее связывание).

Преимущества подхода

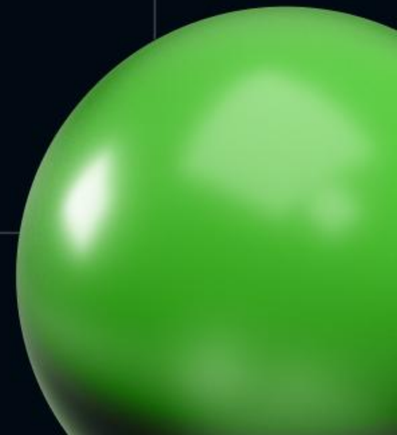
- использование готового и всем знакомого редактора, позволяющего создавать сложные системы иерархий и осуществлять по ним навигацию.



Преимущества подхода

- использование готового и всем знакомого редактора, позволяющего создавать сложные системы иерархий и осуществлять по ним навигацию.

- экономия времени на написание своего UI

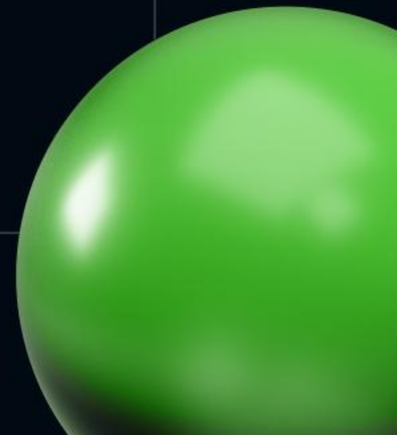


Преимущества подхода

- использование готового и всем знакомого редактора, позволяющего создавать сложные системы иерархий и осуществлять по ним навигацию.

- экономия времени на написание своего UI

- использование c++ кода для описания графического конвейера - не нужны скриптовые языки и виртуальные машины. Свободное использование любой логики, позволяющей настраивать конвейер как угодно



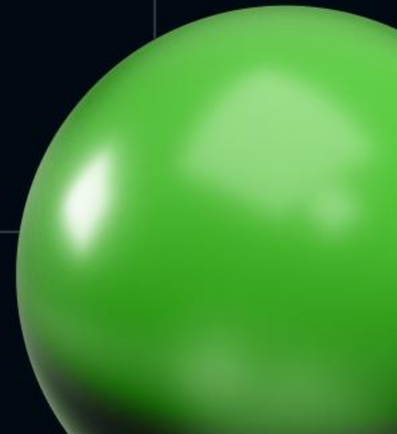
Преимущества подхода

- использование готового и всем знакомого редактора, позволяющего создавать сложные системы иерархий и осуществлять по ним навигацию.

- экономия времени на написание своего UI

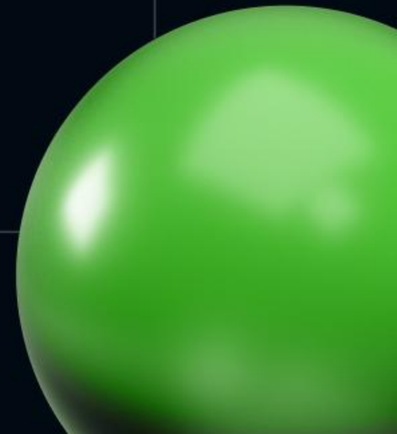
- использование c++ кода для описания графического конвейера - не нужны скриптовые языки и виртуальные машины. Свободное использование любой логики, позволяющей настраивать конвейер как угодно

- интеграция с git “из коробки”



Преимущества подхода

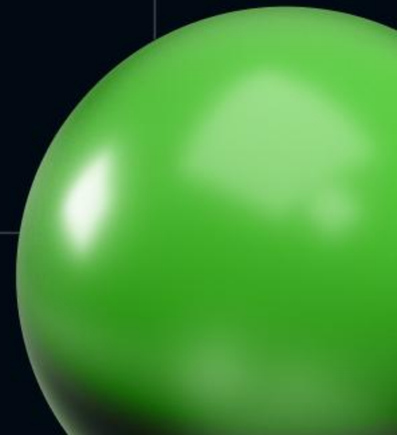
- использование готового и всем знакомого редактора, позволяющего создавать сложные системы иерархий и осуществлять по ним навигацию.
- экономия времени на написание своего UI
- использование c++ кода для описания графического конвейера - не нужны скриптовые языки и виртуальные машины. Свободное использование любой логики, позволяющей настраивать конвейер как угодно
- интеграция с git “из коробки”
- нет проблемы совместимости версий бинарных форматов, не нужна сериализация/десериализация



Недостатки подхода

- Сложные контролы все равно придется писать.

  Но никто не мешает задействовать для них какой-либо GUI сохранив общую логику работы системы.



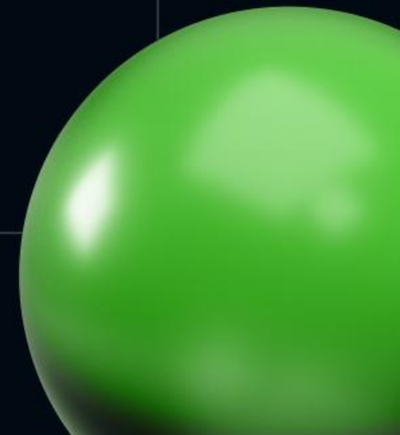
Недостатки подхода

- Сложные контролы все равно придется писать.

  Но никто не мешает задействовать для них какой-либо GUI
  сохранив общую логику работы системы.

- Синтаксическая хрупкость.

  Надо не допускать (через плагин) или откатывать все некорректные
  изменения (например, вставку пустой строки где-либо в c++ коде)
  во время работы движка.



Будущее

- Интеграция с ИИ-агентом.

  Поскольку данные хранятся прямо в c++ коде, т.е. описываются текстом, а поля именованы на манер json, можно добиться управления системой голосом: “сдвинь всех девушек в правую часть сцены”, и т.д.

Будущее

- Интеграция с ИИ-агентом.

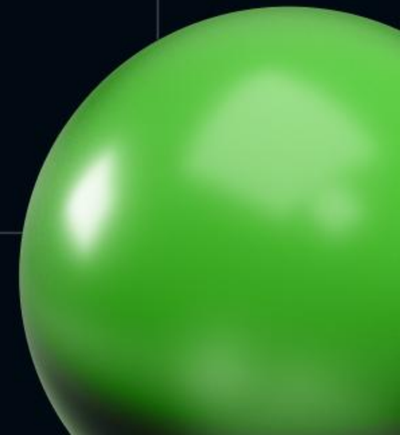
  Поскольку данные хранятся прямо в c++ коде, т.е. описываются текстом, а поля именованы на манер json, можно добиться управления системой голосом: “сдвинь всех девушек в правую часть сцены”, и т.д.

- Графическая навигация.

  Клик по объекту в области просмотра переносит пользователя на нужную строку кода

Для кого?

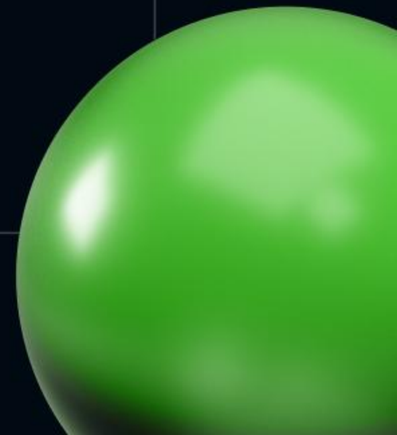
- для прототипирования



Для кого?

- для прототипирования

- для инди-команд сосредоточенных на собственных технологических решениях вместо использования готовых движков

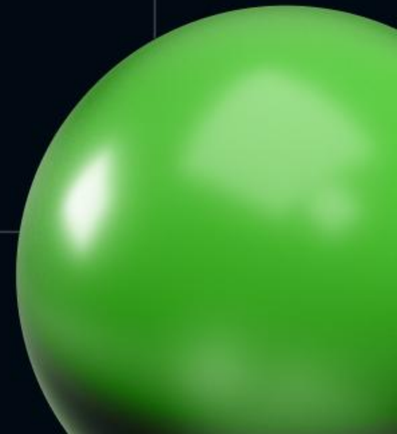


Для кого?

- для прототипирования

- для инди-команд сосредоточенных на собственных технологических решениях вместо использования готовых движков

- для всех, у кого возникает необходимость удобной настройки параметров напрямую из кода - к примеру, когда UI еще не написан



Спасибо за внимание!

репозиторий проекта (прототип WIP)

- В ближайших планах - отделить функционал от движка в самостоятельную библиотеку.
- Опенсорс. Критика, предложения, коммиты принимаются с благодарностью

трейлер игры The 13th Sign, сделанный на этом движке

вопросы можно задать в Телеграм: @lastshilling

Вопросы?