Многопоточность против фризов

FROSTGATE
STUDIO

Артамонов В.В.

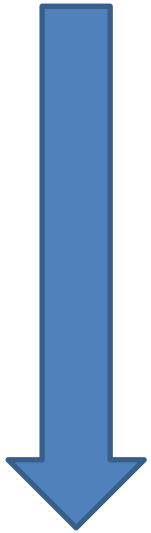
Лаг при загрузке

Лаг при вводе (input lag)

Лаг при вводе (input lag)

Нормальная нагрузка

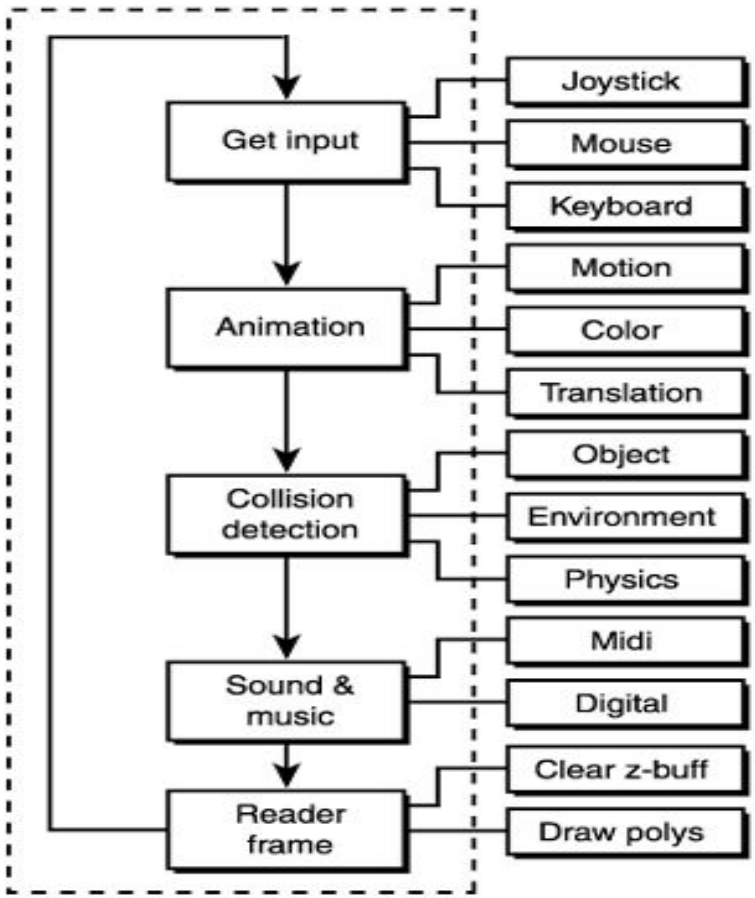
Задачи

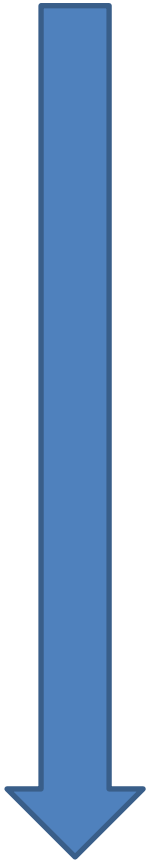


Перегрузка

Много задач

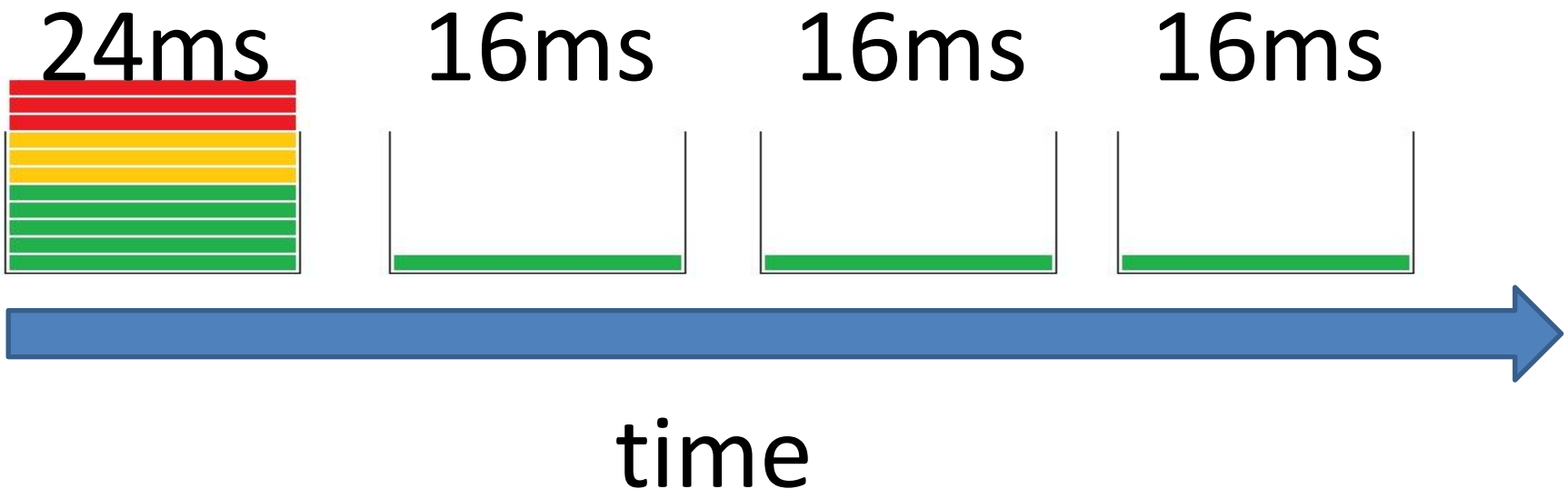
Игровой цикл



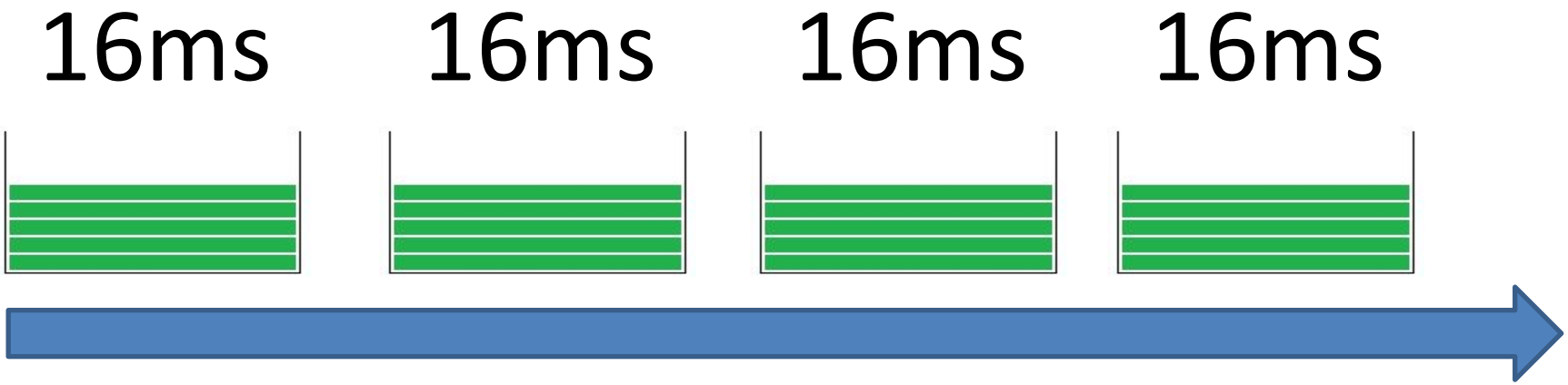


16ms

Фриз при пиковой нагрузке



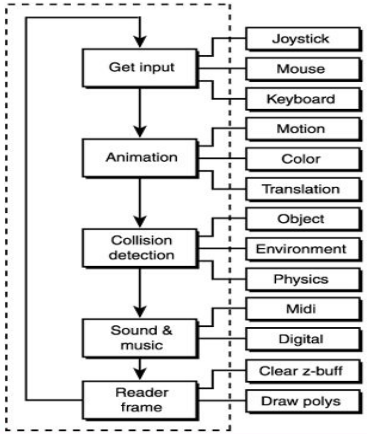
Корутины

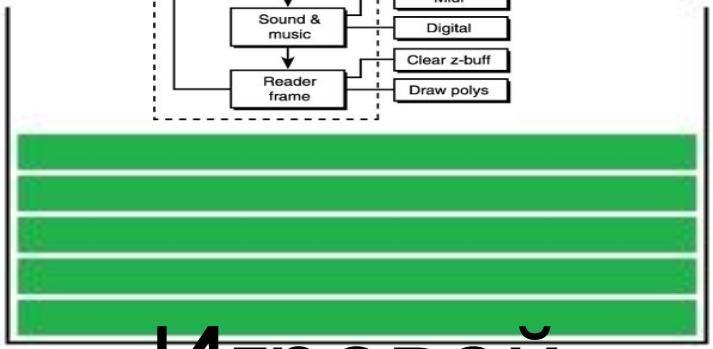


Недостатки корутин

- Однопоточность
- Нет возвращаемого значения
- Сложно обрабатывать ошибки

Многопоточность





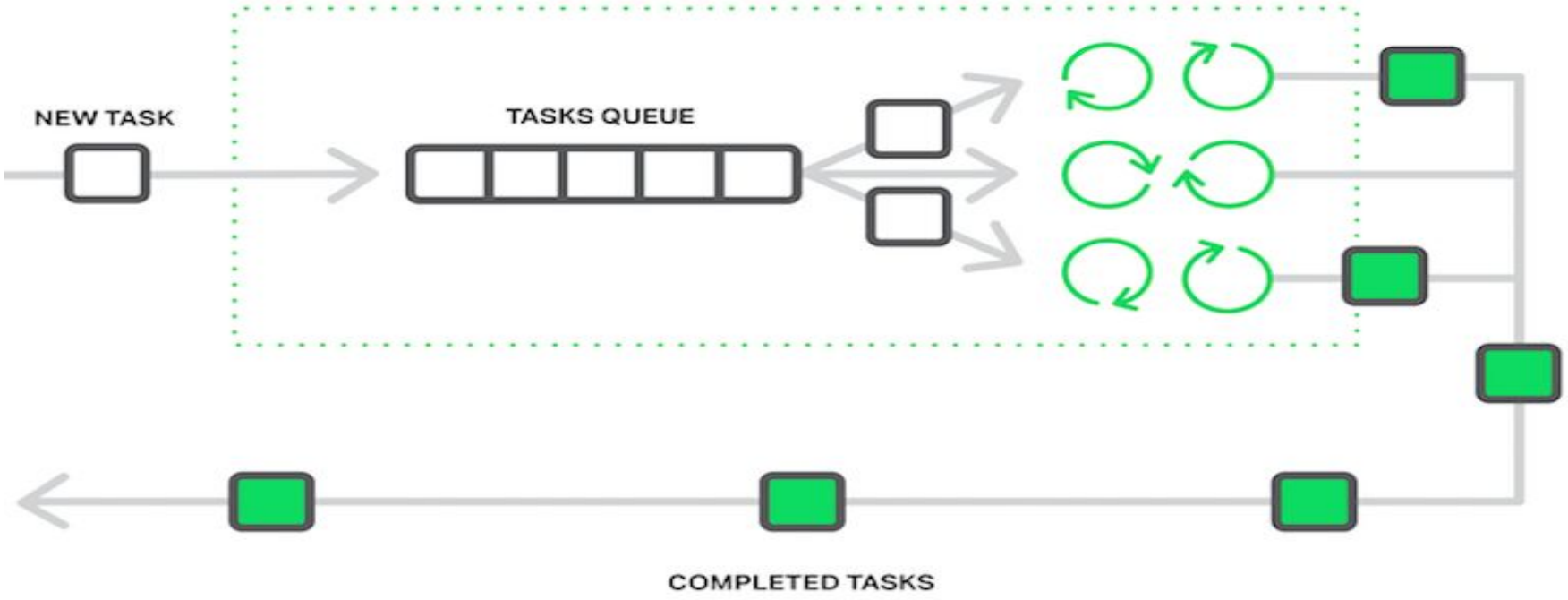
Игровой цикл

Thread 1

Thread 2

Thread 3

Пул потоков (ThreadPool)



Пул потоков (ThreadPool)

```
void Awake()
{
    // Unity main thread

    ThreadPool.QueueUserWorkItem(q => {
        // Background thread
    });

    // Unity main thread
}
```

Асинхронные вычисления

```
async void Awake()
{
    // Unity main thread

    await Task.Run(() =>
    {
        // Background thread
    });

    // Unity main thread
}
```

Асинхронный ввод-вывод

```
HttpClient client;

async void Awake()
{
    // Unity main thread

    string json = await client.GetStringAsync("http://...");

    // Unity main thread
}
```

Цепочка асинхронных методов

```
async Task DoSomethingAsync()
{
    await DoSomethingElseAsync();
}

async Task DoSomethingElseAsync()
{
    await Task.Delay(1000);
}
```

Асинхронный возврат значения

```
async Task DoSomethingAsync()
{
    int result = await DoSomethingElseAsync();
}

async Task<int> DoSomethingElseAsync()
{
    await Task.Delay(1000);
    return 25;
}
```

Асинхронная обработка ошибок

```
async Task DoSomethingAsync()
{
    var task = DoSomethingElseAsync();

    try
    {
        await task;
    }
    catch (Exception e)
    {
        // do something
    }
}
```

```
async Task DoSomethingElseAsync()
{
    throw new Exception();
}
```

Асинхронность вместо корутин

```
IEnumerator Start()
{
    Debug.Log("Waiting 1 second...");
    yield return new WaitForSeconds(1.0f);
    Debug.Log("Done!");
}

async void Start()
{
    Debug.Log("Waiting 1 second...");
    await Task.Delay(TimeSpan.FromSeconds(1));
    Debug.Log("Done!");
}
```

Асинхронность вместо корутин

```
IEnumerator Start()
{
    yield return Resources.LoadAsync("...");
}

async void Start()
{
    await Resources.LoadAsync("...");
}
```

Настройка Unity

Edit -> Project Settings -> Player

- Scripting Runtime Version .NET 4.x Equivalent
- Api Compatibility Level .net standard 2.0