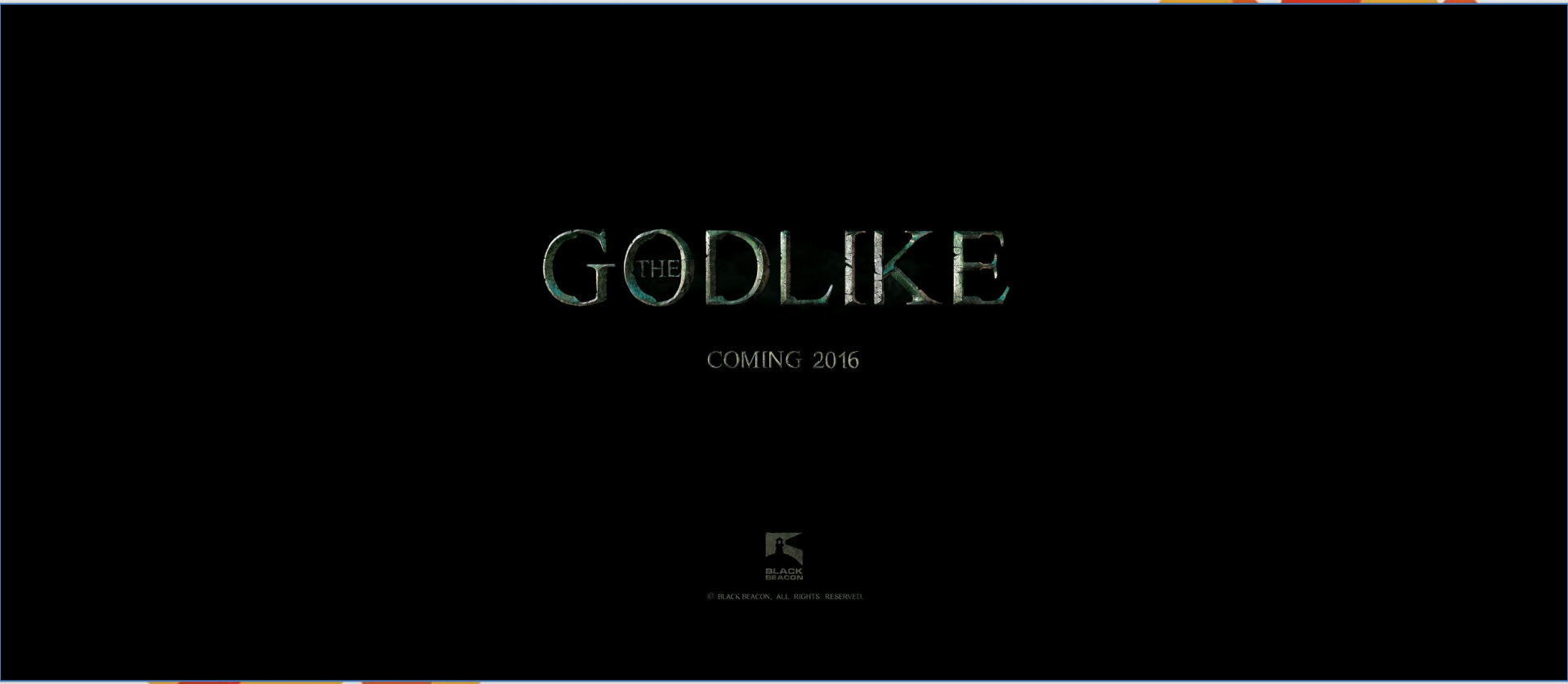
Разработка крупного standalone проекта на Unity

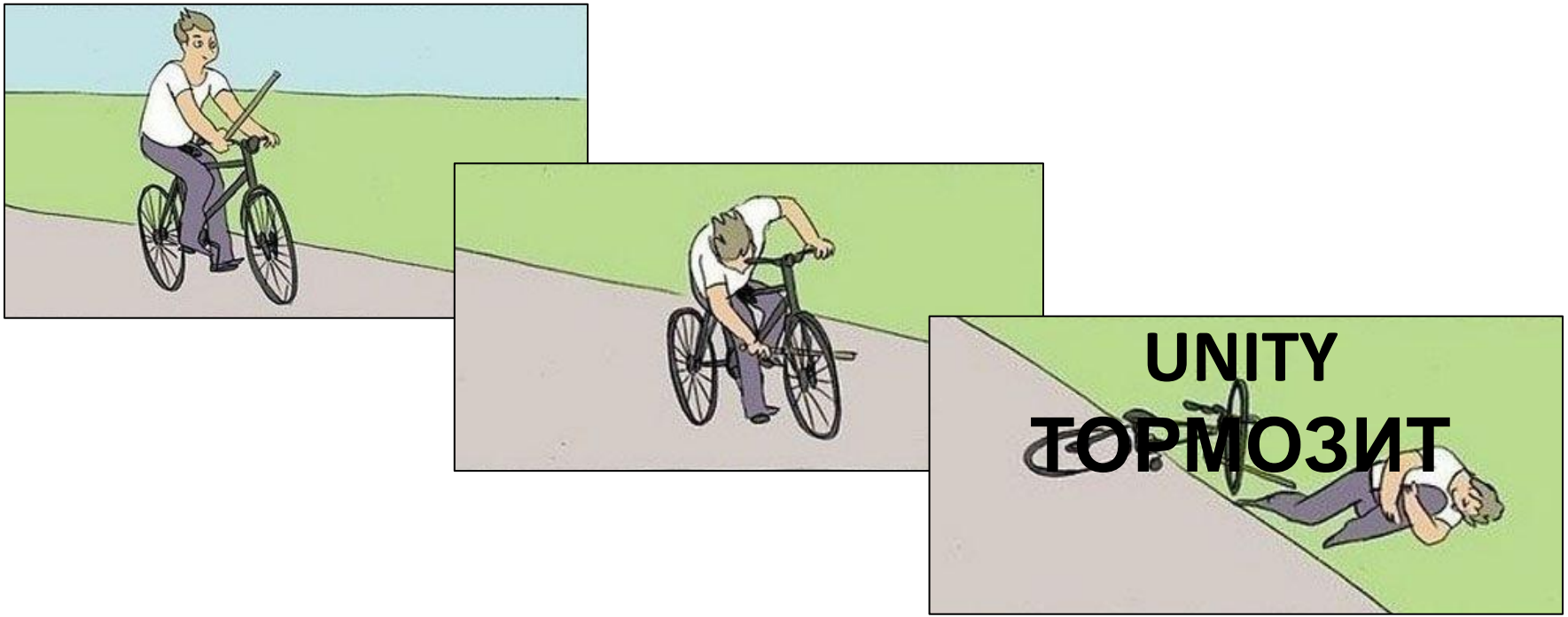
улучшаем производительность

Кто мы такие

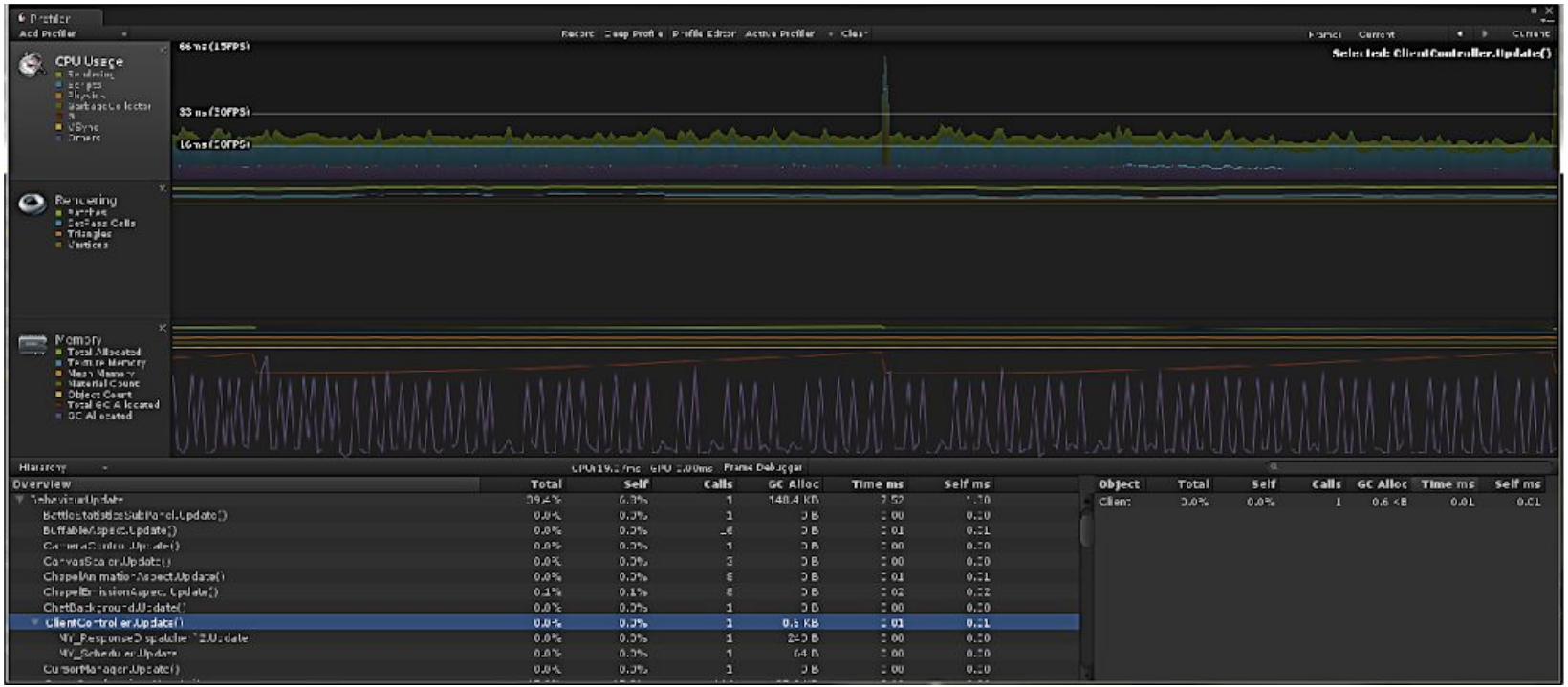

THE GODLIKE
COMING 2016
BLACK BEACON
© BLACK BEACON. ALL RIGHTS RESERVED.

- Клиентская игра под Windows и Mac
- Авторитарный сервер, тоже на Unity
- Игровая сцена = 2000+ объектов со скриптами
- Команда ~ 40 человек

Оптимизация



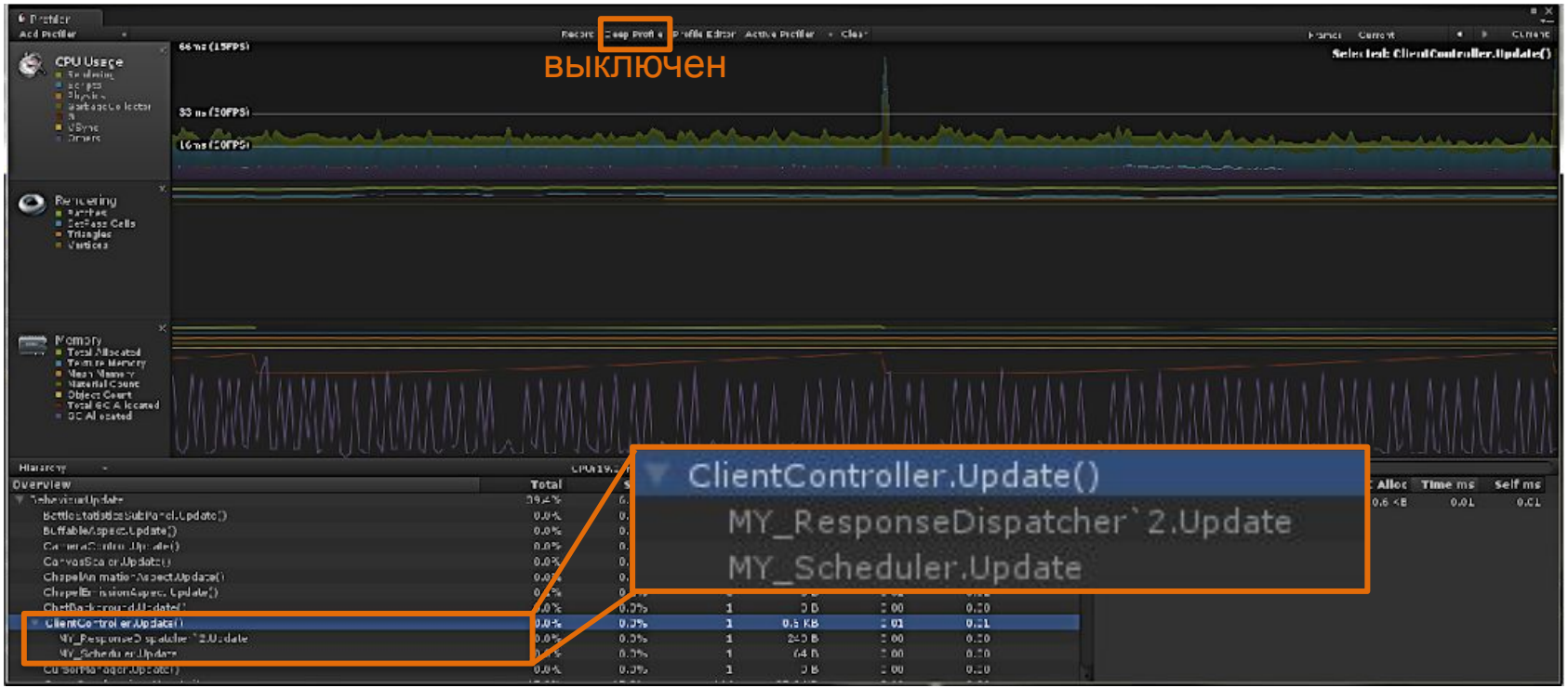
Профайлер


Profiler
Add Profiler
Record
Deep Profile
Profile Editor
Active Profiler
Clear
CPU Usage
Rendering
Memory
Hierarchy
Overview
Total
Self
Calls
GC Alloc
Time ms
Self ms
Object

Профайлер

```
[UsedImplicitly]
private void Update()
{
     var updatablesCount = updatableObjects.Count;
     for (int i = 0; i < updatablesCount; i++)
     {
     var item = updatableObjects[i];
          Profiler.BeginSample("MY_"+item.GetType().Name+".Update");
          item.Update();
          Profiler.EndSample();
     }
}
```

Профайлер


выключен
ClientController.Update()
MY_ResponseDispatcher`2.Update
MY_Scheduler.Update

Профайлить в файл

```
private void Update()
{
      if (Time.frameCount % 300 == 1)
      {
            var fileName = GenerateFileName();
            Profiler.enabled = false; //Освобождаем старый фай
            var logFile = Path.Combine(folder.FullName, fileName)
            Profiler.logFile = logFile;
            Profiler.enabled = true; //Создаем новый файл
      }
}
```

http://pastebin.com/ktVgrxyq

Профайлить в файл

Результат

T:

_15601.prf	data	768 300	09.12.2015 13:51
_15601	prf	2 239	09.12.2015 13:51
_15301.prf	data	5 125 620	09.12.2015 13:51
_15301	prf	15 490	09.12.2015 13:51
_15001.prf	data	5 130 896	09.12.2015 13:51
_15001	prf	15 491	09.12.2015 13:51
_14701.prf	data	5 133 528	09.12.2015 13:51
_14701	prf	15 492	09.12.2015 13:51
_14401.prf	data	5 124 680	09.12.2015 13:51
_14401	prf	15 496	09.12.2015 13:51
_14101.prf	data	5 120 340	09.12.2015 13:50
_14101	prf	15 495	09.12.2015 13:50
_13801.prf	data	5 039 076	09.12.2015 13:50
_13801	prf	15 491	09.12.2015 13:50

http://pastebin.com/ktVgrxyq

Профайлить в файл

```
[ContextMenu("Load folder")]
private void Update()
{
	path = EditorUtility.OpenFolderPanel("profiler", "", "");
	allFiles = Directory.GetFiles(path, "*.prf");
	currentFile = PickFile();
	Profiler.AddFramesFromFile(allFiles[currentFileId]);
}
```

http://pastebin.com/ktVgrxyq

GarbageCollector

- Вызывается в любое удобное **ему** время
- GC.Collect 
- Поколения GC 

Источники мусора

Строки

BAD

```
Var s = “A”+”B”+”C”+”D”;
```

```
String.Format(“{0}{1}{2}{3}”,
                A,B,C,D);
```

```
Void Update(){
      text = SomeField.ToString();
}
```

GOOD

```
StringBuilder.Length = 0;
StringBuilder.Append(‘A’)
                .Append(‘b’)
                .Append(‘c’)
                .Append(‘d’);
```

```
Int SomeField
{
       get{return_inner;}
       set{
             if(field=value) return;
             text = SomeField.ToString();
          }
}
```

Источники мусора

Замыкани

BAD

```
Void Subscribe()
{
      Var stuff = GetStuff();
      other.OnSomeEvent +=
            t => DoSomething(Stuff)
}

Void DoSomething(T t, Tstuff stuff)
{
      stuff.Do(t);
}
```

GOOD

я

```
Void Subscribe()
{
      other.OnSomeEvent +=
            t => DoSomething(t)
}

Void DoSomething(T t, Tstuff stuff)
{
      Var stuff = GetStuff();
      stuff.Do(t);
}
```

Источники мусора

LINQ

BAD

```
List<T> selection;

Void OnShiftDragGropped()
{
    selection = selection
        .Union(newSelection) //ext
        .Distinct()
        .ToList();           //new
}
```

GOOD

```
List<T> selection;

Void OnShiftDragGropped()
{
    foreach(var item in newSelection)
    {
        if(!selection.Contains(item)
            {selection.Add(item)
    }
}
```

BEST

```
Hashset<T> selection;
```

Кодогенерация

SourceAssembly.dll

T4

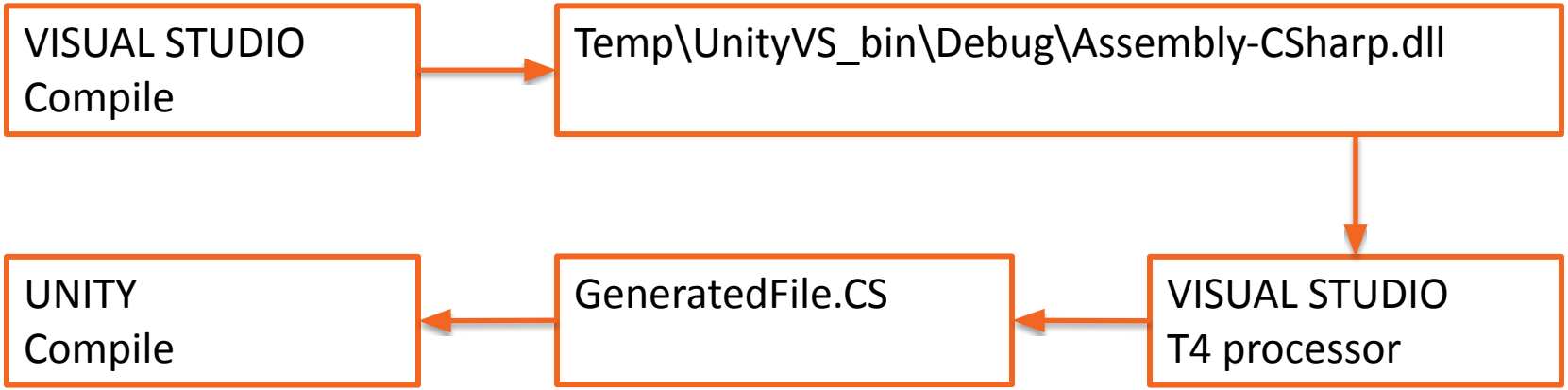
GeneratedCode.CS

SourceAssembly.dll

Mono.Cesil

AugmentedAssembly.dll

Кодогенерация T4



T4 HowTo

```
<#@ template debug="true" hostspecific="false" language="C#" #>
<#@ output extension="cs" #>
<#@ assembly name="$(SolutionDir)\Temp\UnityVS_bin\Debug\Assembly-CSharp.dll" #>
<#@ import namespace="System" #>
<#@ import namespace="System.Reflection" #>
<#@ import namespace="SoH.Common.DataPresentation.Curve.CodeGeneration" #>
using System;
using System.Collections.Generic;
using SoH.Common.BitStream.DataTypes;
```

<#@ Директивы процессора #>

T4 HowTo

```
partial void CreateUpdaters(object component, List<ICurveUpdater> result)
{
<#    foreach(Type t in classes)
      {
#>
      if (component is <#=t.Name#>){ result.AddRange(CreateUpdaters((<#=t.Name#>)component));
return;}
<#
      }
#>
      throw new InvalidOperationException(component.GetType().Name
                                          + "should have [GenerateCurves] attribute");
}
```

<# код который выполнит шаблон #>

<#= переменная, значение которой подставится в результат #>

T4 HowTo – сгенерированный код

```
partial void CreateUpdaters(object component, List<ICurveUpdater> result)

{
    if (component is BattleStatistic)              { result.AddRange(CreateUpdaters((BattleStatistic)component));          return;}
    if (component is BuffableAspect)       { result.AddRange(CreateUpdaters((BuffableAspect)component));           return;}
    if (component is BuildingTargetAspect)         { result.AddRange(CreateUpdaters((BuildingTargetAspect)component));     return;}
    if (component is DoubleProgressBarAspect)      { result.AddRange(CreateUpdaters((DoubleProgressBarAspect)component)); return;}
    if (component is FogObjectAspect)              { result.AddRange(CreateUpdaters((FogObjectAspect)component));          return;}
    if (component is InfluenceContainer)           { result.AddRange(CreateUpdaters((InfluenceContainer)component));       return;}
    if (component is InteractionAspect)            { result.AddRange(CreateUpdaters((InteractionAspect)component)); return;}
    if (component is Outpost)              { result.AddRange(CreateUpdaters((Outpost)component));                  return;}
    if (component is Pingometer)                   { result.AddRange(CreateUpdaters((Pingometer)component));        return;}
    if (component is PlayerResources)              { result.AddRange(CreateUpdaters((PlayerResources)component));          return;}
    if (component is Projectile)                   { result.AddRange(CreateUpdaters((Projectile)component));        return;}
    if (component is ProjectileNetworkChannel)     { result.AddRange(CreateUpdaters((ProjectileNetworkChannel)component)); return;}
    if (component is ShopAspect)                   { result.AddRange(CreateUpdaters((ShopAspect)component));        return;}
    if (component is SideChangerAspect)            { result.AddRange(CreateUpdaters((SideChangerAspect)component)); return;}
    if (component is Squad)                        { result.AddRange(CreateUpdaters((Squad)component));                    return;}
    if (component is SquadTargetAspect)            { result.AddRange(CreateUpdaters((SquadTargetAspect)component)); return;}
    if (component is UndeadBuildingAspect)         { result.AddRange(CreateUpdaters((UndeadBuildingAspect)component));     return;}

    throw new InvalidOperationException(component.GetType().Name + "should have [GenerateCurves] attribute");
}
```

T4 HowTo – сгенерированный код

Самые популярные

- `List<T>`,
- `T[]`
- `Dictionary<string, T>`

А есть еще:

- `HashSet<T>` и `SortedSet<T>`
- `SortedList<TKey, TValue>`
- `SortedDictionary<TKey, TValue>`
- `LinkedList<T>`
- `Stack<T>` и `Queue<T>`

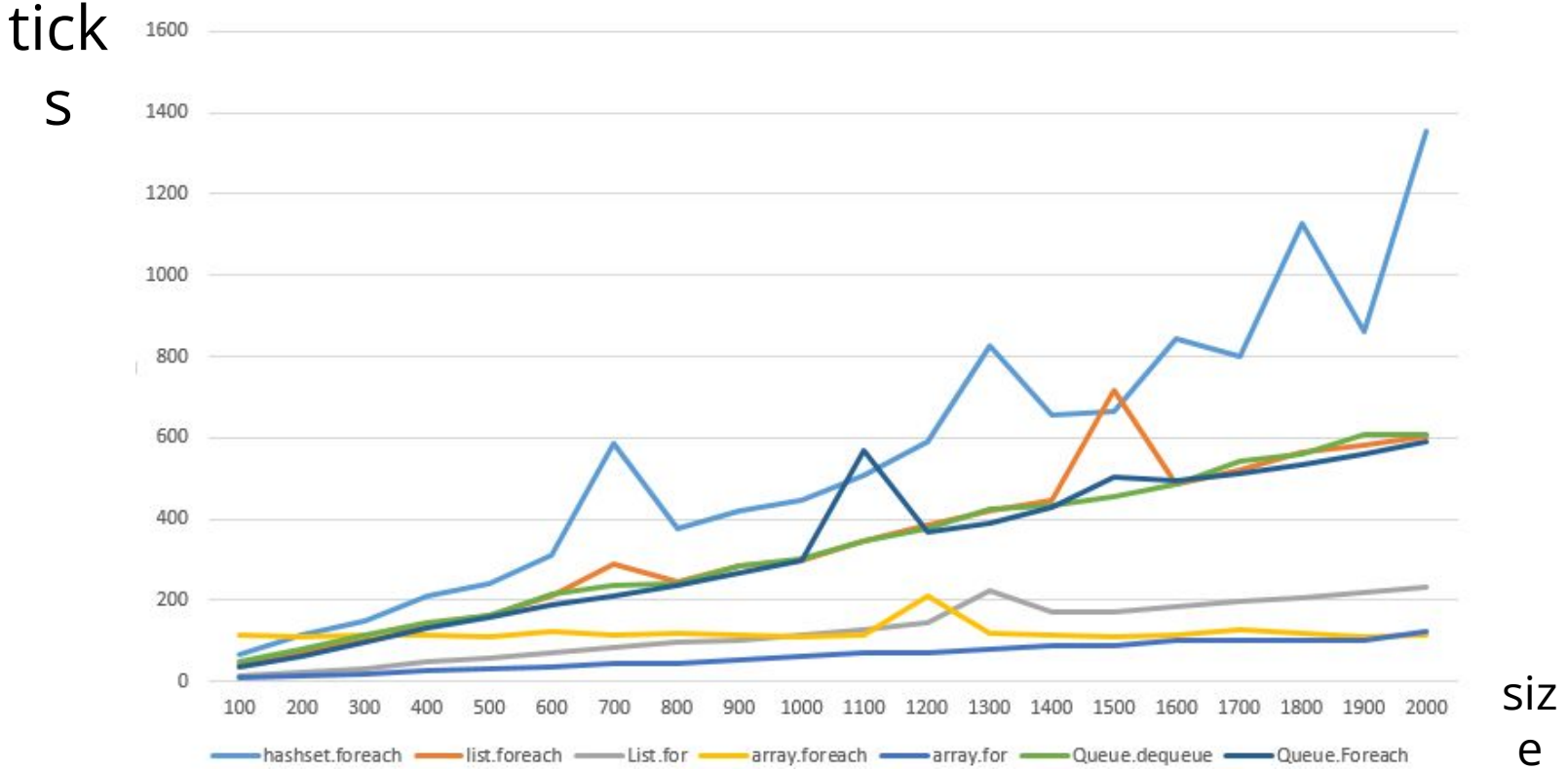
Умные структуры данных и тупой код работают куда лучше, чем наоборот. // E. Raymond

Collections time

	Remove	Add	Contains	Iterate
List	O(n)	O(n)	O(n)	FAST
Dictionary	O(1)	O(1)	O(1)/O(*n*)	FAST
HashSet	O(1)	O(1)	O(1)	SLOW
SortedSet	O(log *n*)	O(log *n*)	O(1)	SLOW
SortedList	O(n)	O(n)	O(log *n*)	FAST
SortedDictionary	O(log *n*)	O(log *n*)	O(1)/O(*n*)	SLOW
Stack/Queue	O(1)*	O(1)	O(n)	FAST

* Удаляет элемент из коллекции

Collections time



Еще советы

- Кэшируйте все подряд. GO, промежуточные вычисления, ссылки на монобехи.
- Там, где вам не нужен расчет в 3D – не делайте его
- Не делайте лишних операций с векторами(структурами), пользуйтесь конструктором
- Не используйте Debug.Log без надобности
- **If** работает быстрее чем хитрая формула. Хитрые формулы оставте для шейдеров
- Пользуйтесь асинхронными операциями LoadLevelAsync и Resources.LoadAsync
- Пользуйтесь быстрыми проверками чтобы отсеить ненужное: AABB или евклидово расстояние
- Пользуйтесь Buffer.Copy для быстрого копирования

Спасибо за внимание