


游戏引擎技术点滴

从五个商业引擎的沉浮说起

Game Engine Topics

 十年变迁 (2006-2016)

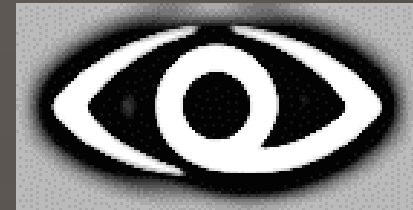
 一次技术迭代周期

前期：引擎的评估和技术选型

后期：引擎的运用，技术改造和问题解决

 下一个十年

Game Engines

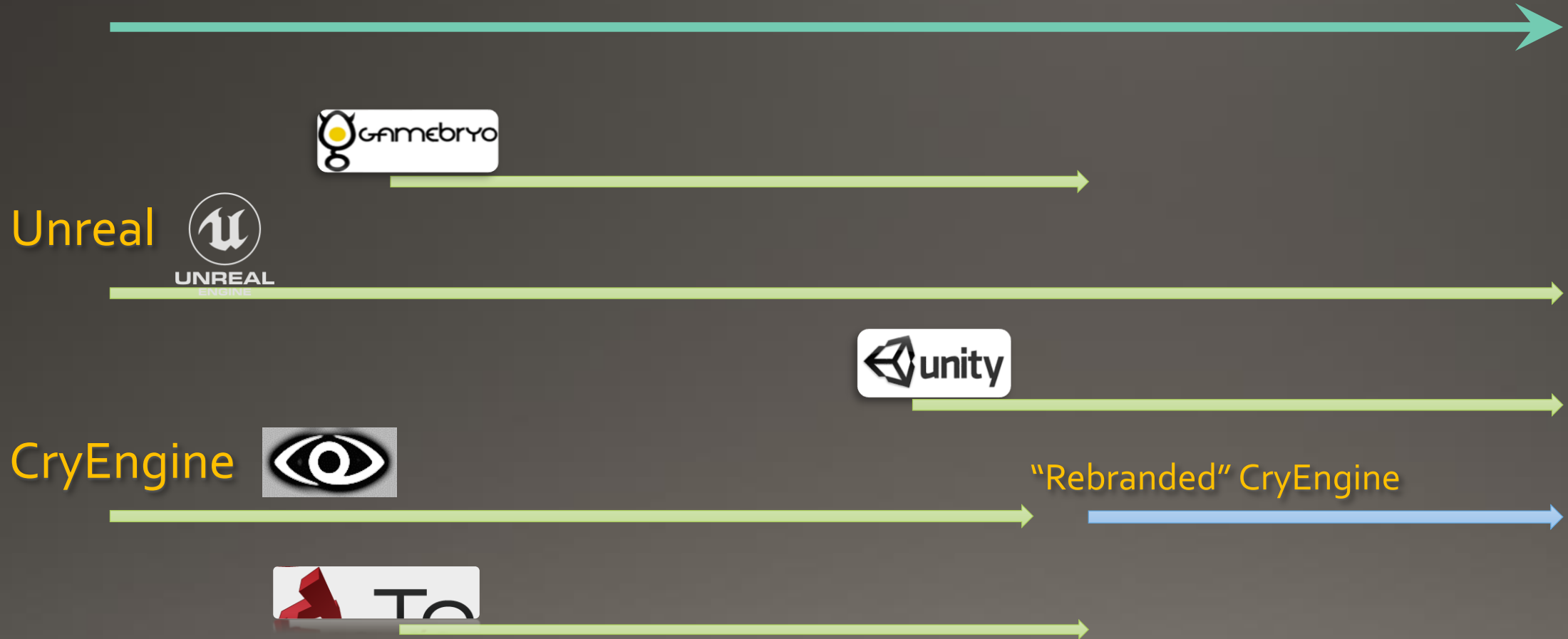


~~OGRE / Irrlicht / cocos2d-x~~

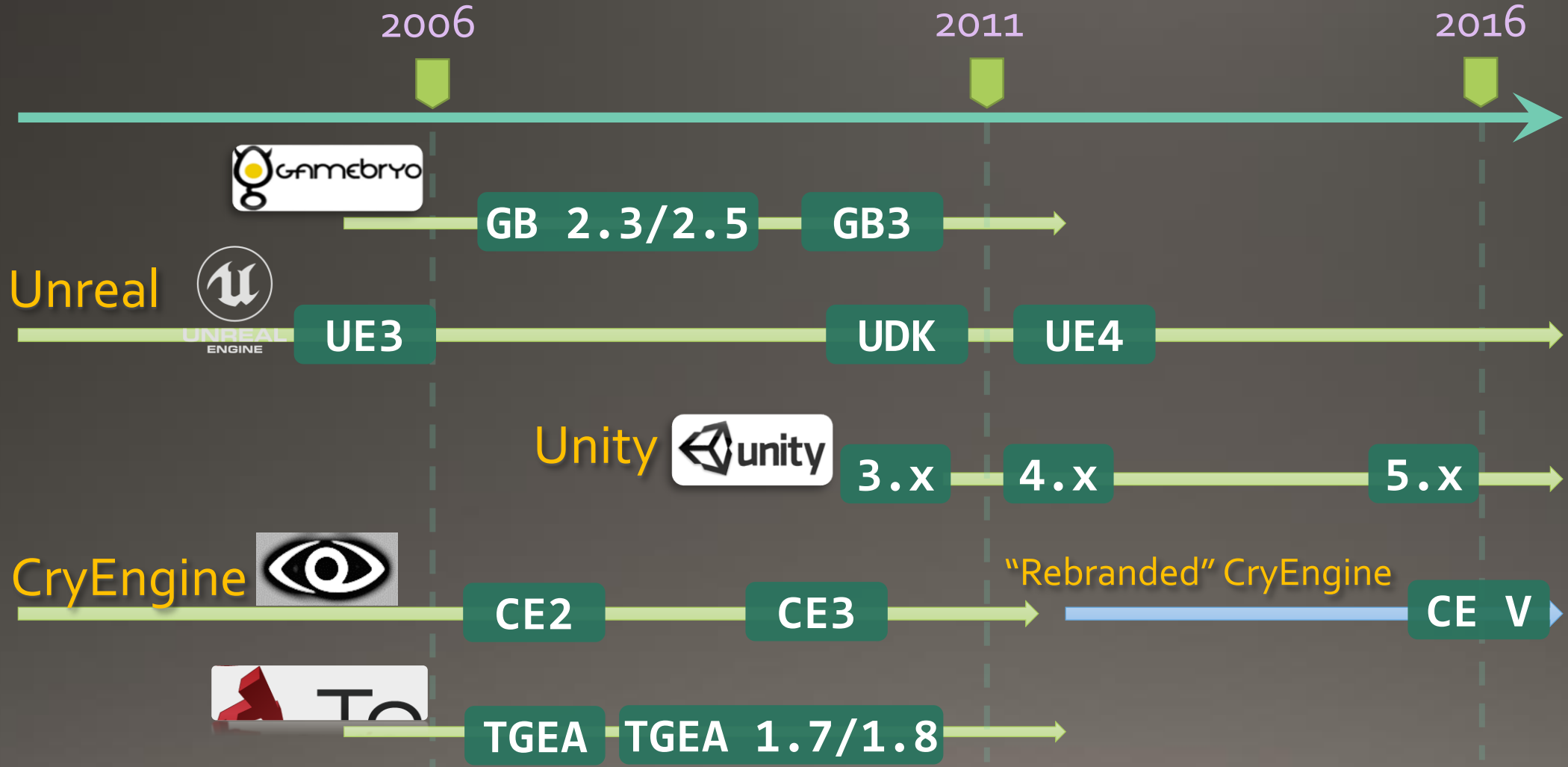
Timeline (1)

2006

2016

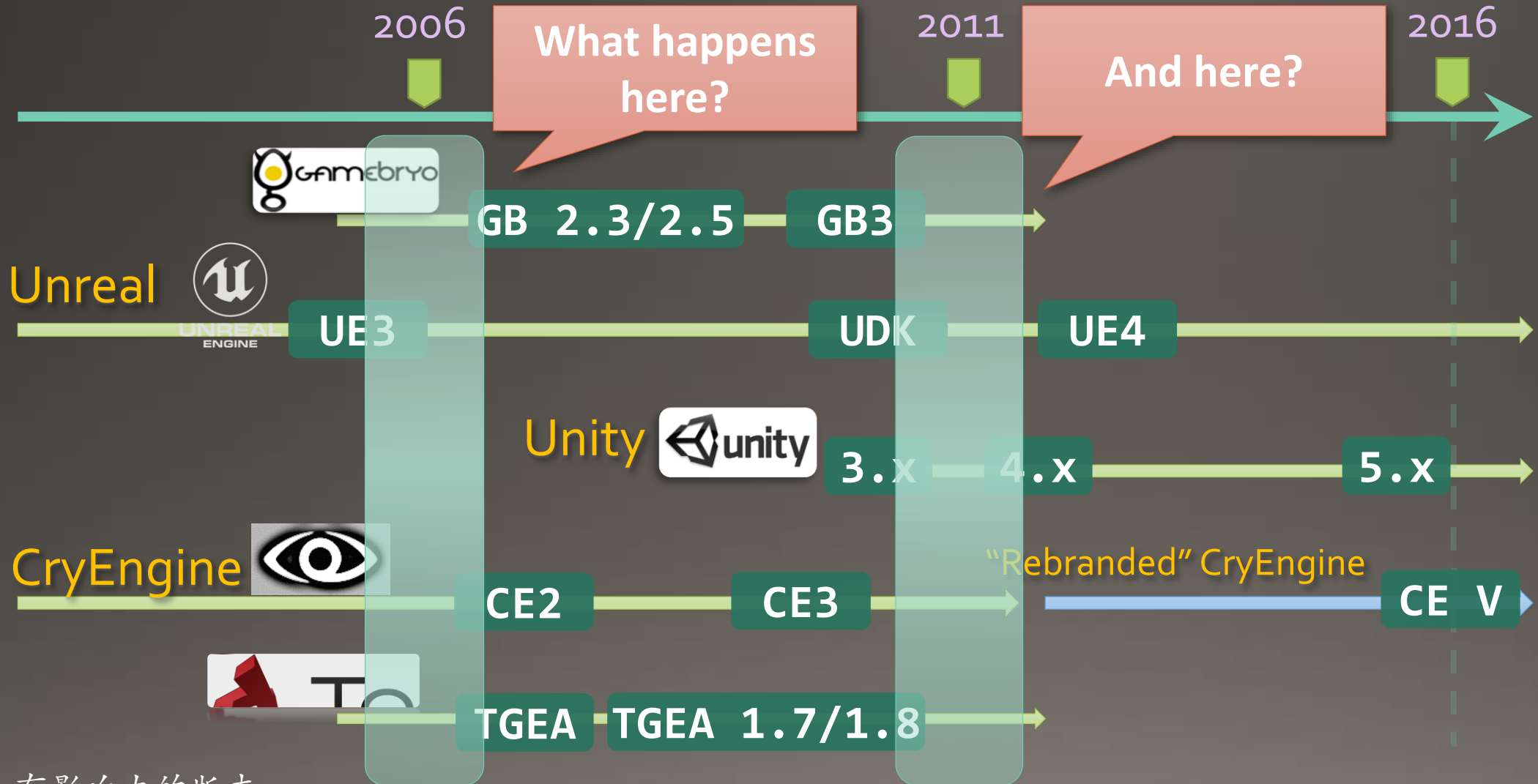


Timeline (2) Milestones



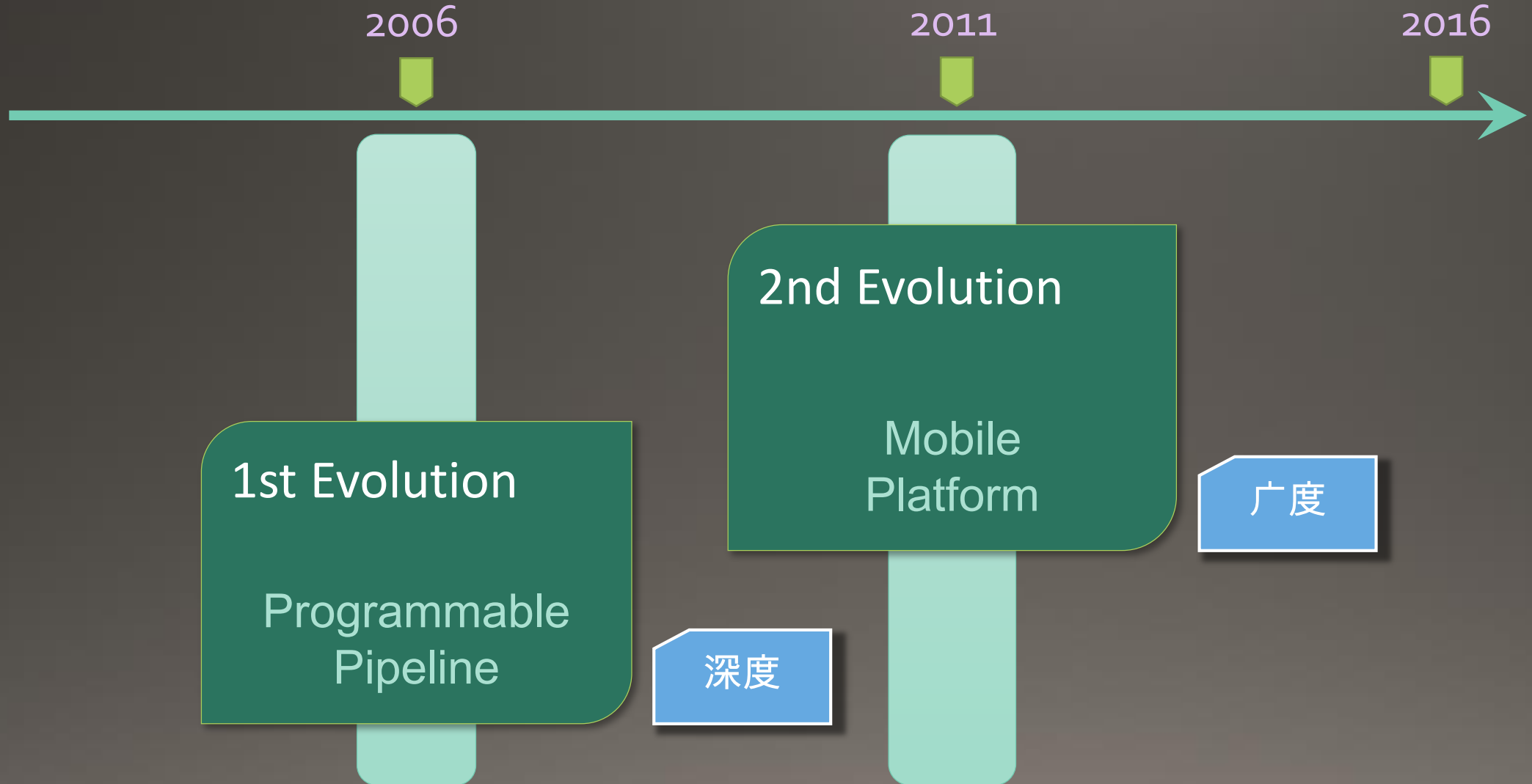
Ver. 有影响力的版本

Timeline (3)

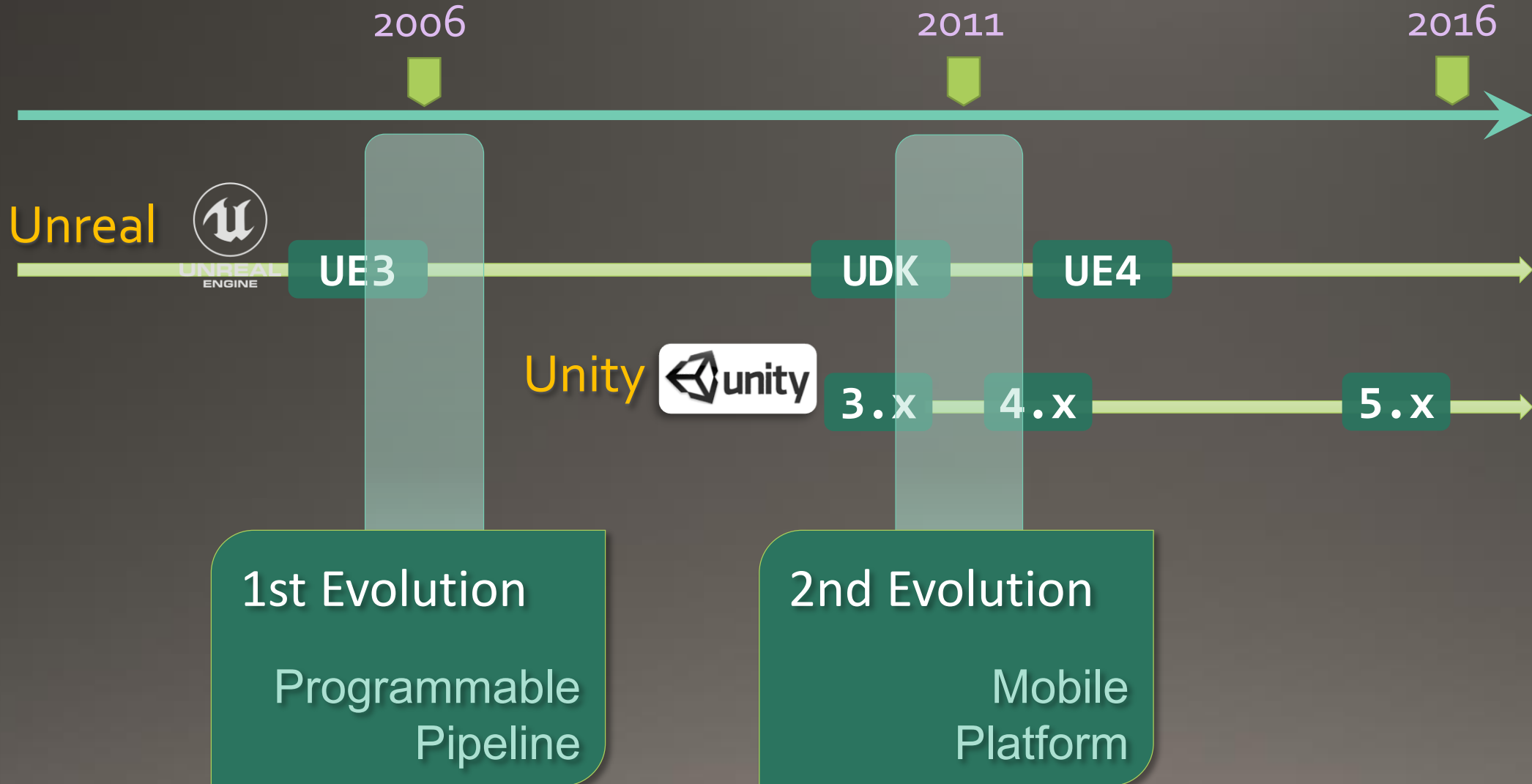


Ver. 有影响力的版本

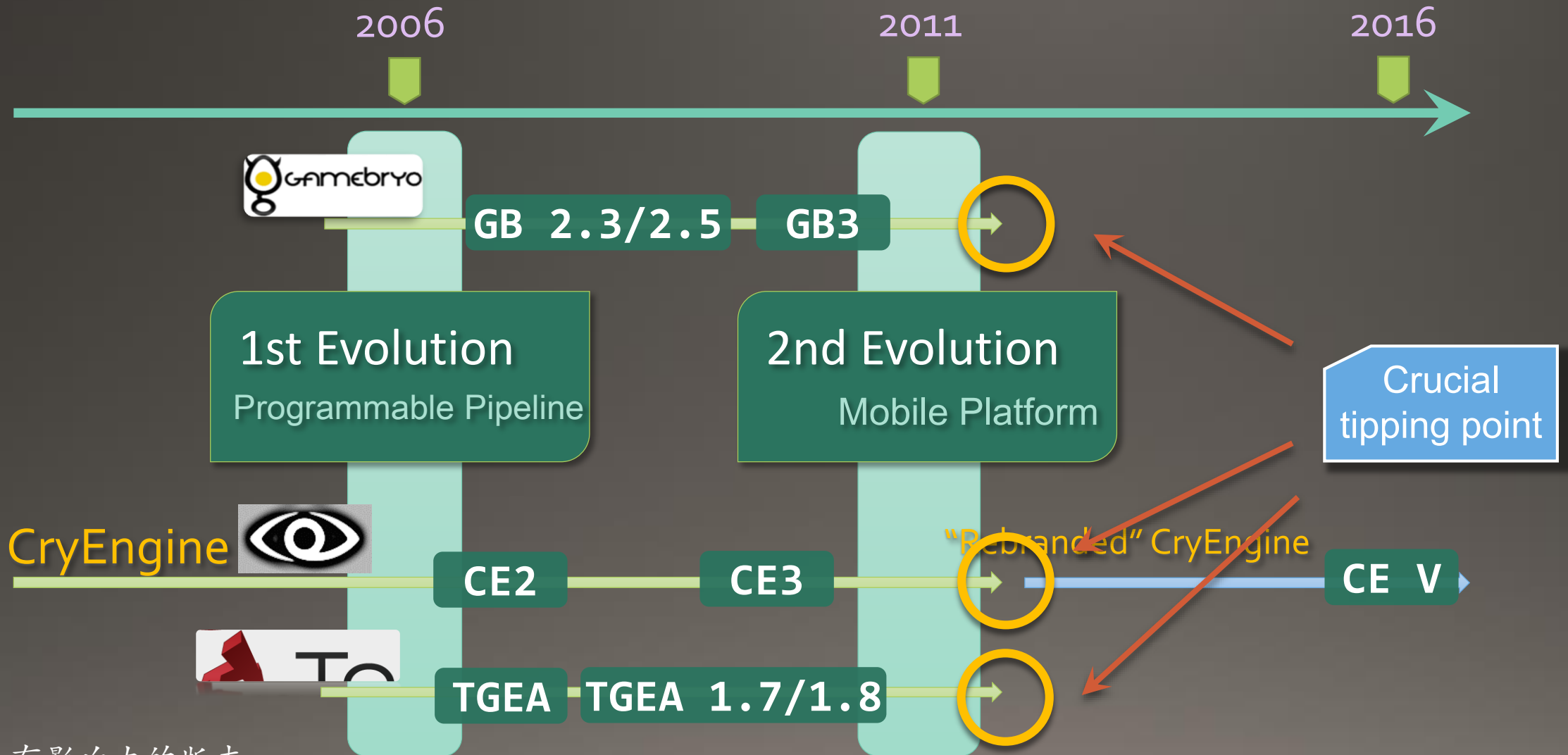
Timeline (4) Industry Evolutions



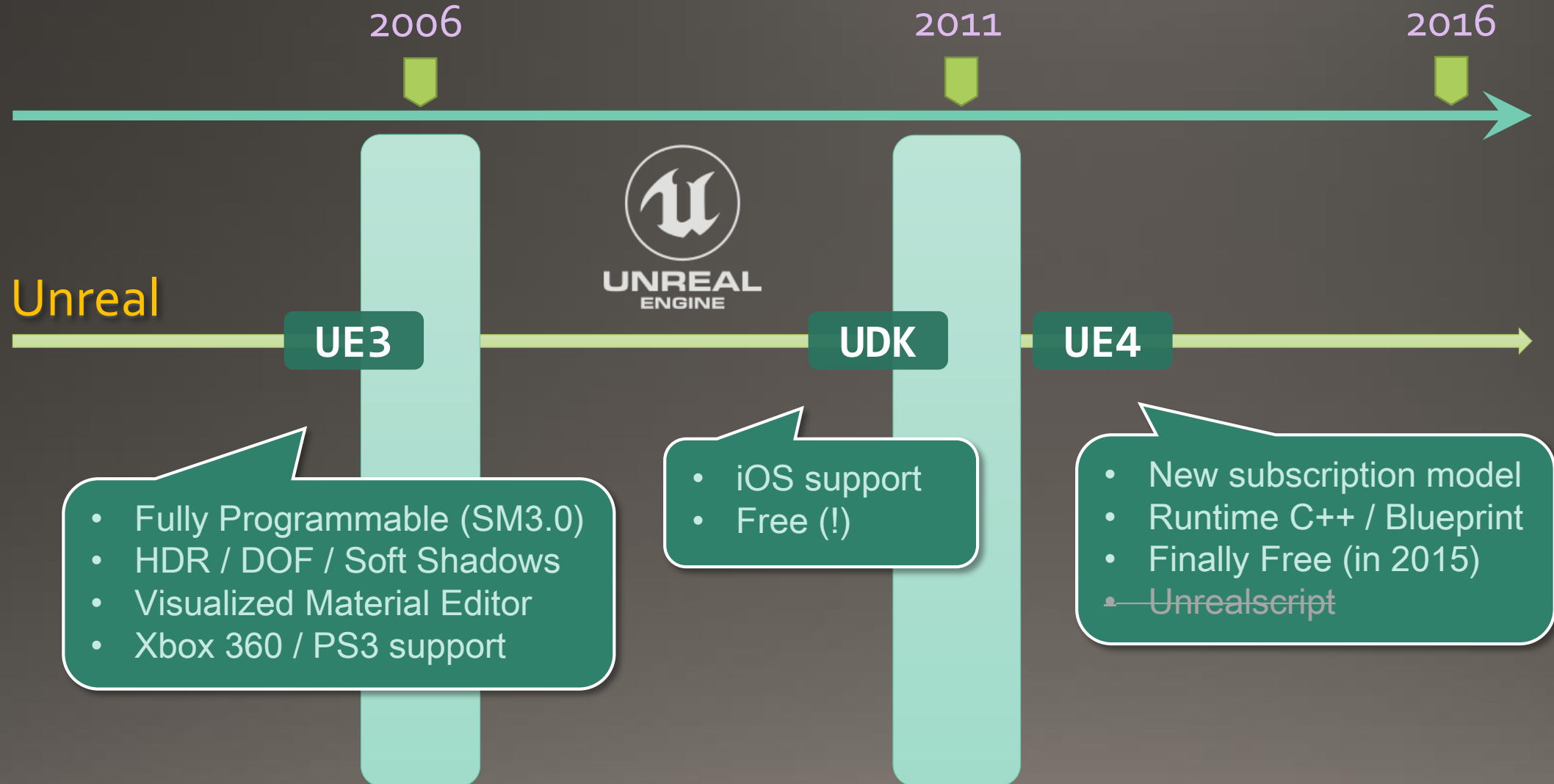
Timeline (5) Some of them



Timeline (6) And the rest of them



Timeline (7) What did it do?



Conclusion

不是最强的，也不是最新最酷的，更不是最贵的，而是最适应变化的，活了下来。

It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change.

- Charles Darwin

Game Engine Topics

✓ 十年变迁 (2006-2016)

一次技术迭代周期

前期：引擎的评估和技术选型

后期：引擎的运用，技术改造和问题解决

• 下一个十年

评估 - 三个必问问题

- ✎ 是否有经过验证的（同品类）产品？
- ✎ 好招人吗？
- ✎ 是否有完整的代码？

注：这几个问题并非一定需要肯定的答案，但如果任何一项答案是“否”，就需要审慎考虑可能带来的影响。

评估 - 定义引擎和游戏的关系

类库式 (classlibrary-style)

受控

- 低耦合，接触面小，对游戏约束小，升级负担轻
- 间接层带来的开发和运行效率的损失

框架式 (framework-style)

快!

- 高度相互依赖，当引擎有较大更新时适配工作量过大
- 从原型到产品速度快

注：技术基调选择，没有对错，需要做出与团队综合能力，企业文化等相匹配的选择。

评估 - 考虑因素(一级)

- ✎ feature set - 特性集与需求的匹配度
- ✎ tool chain completeness - 工具链完备性
- ✎ minimum iteration time - 最小迭代时间

注：这些因素直接决定了项目按期交付的可能性。

评估 - 考虑因素(二级)

exception tolerance - 异常耐受力

- 特殊需求
- 坏数据
- 破坏性改动

multi-dimensional visibility - 充分的可见性

- 关键业务流程的可见性
- 文件数据可见性
- 性能数据可见性

注：这些因素直接决定了风险把握和问题解决的可能性。

评估 - 考虑因素(三级)

- ✎ workflow customizability - 工作流的可定制性
- ✎ teamscale-friendly - 对团队规模变化是否友好
- ✎ 3rd party-friendly - 对第三方库是否友好
- ✎ industry standard-friendly - 对行业标准是否友好
- ✎ community-friendly - 社区问题解答能力，互相作用的方式

三不管地带

运用和改造

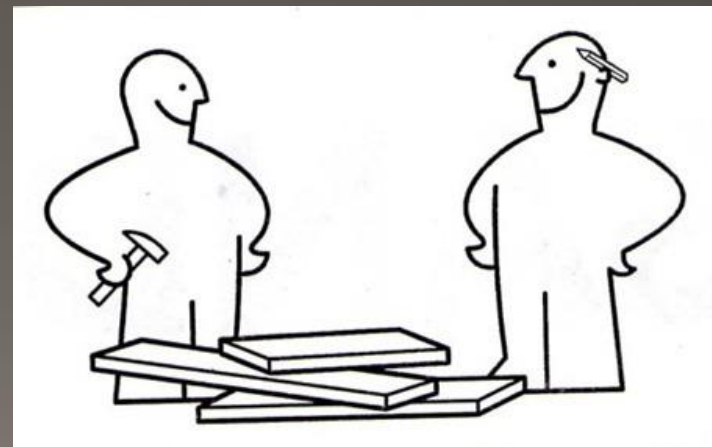
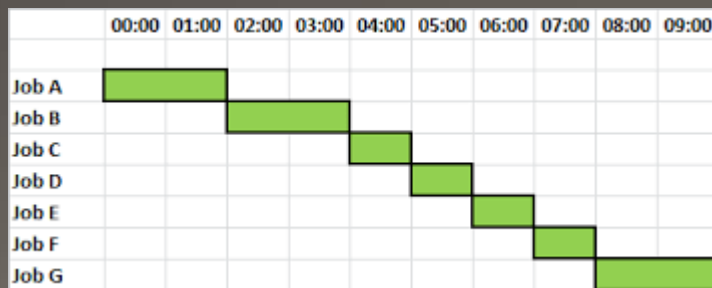
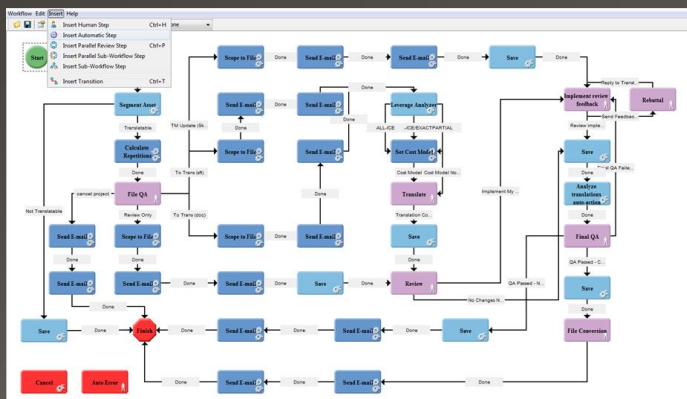
🖍 工作流的建立，维护和优化

🖍 注意观察工作流中每一环的个体响应速度

🖍 注意清理不必要的依赖关系（逻辑上/资源上/人员上）

🖍 注意区分 hackers 和 scientists，总是让他们待在合适的地方

心跳依赖



引擎版本同步节奏的管理

- ✎ 引擎版本同步就像持续构建 (Continuous Integration)，做得越及时，粒度越细，总的工作量就越小
- ✎ 每次新版本的集成，都是...
 - 一次检视各个子模块的潜在缺陷的好机会
 - 一次发现不同的自动化测试点的好机会
 - 一次清理负担，轻装上阵的机会
- ✎ 应总是在**实验分支**里做集成，这样就可以在日后回溯你的每一次集成的完整经过，降低集成时新引入问题的风险

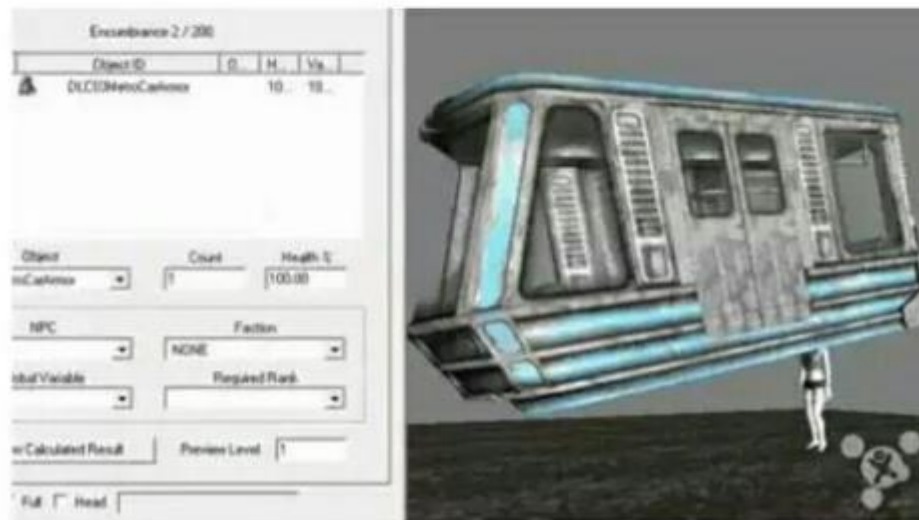
局限性认识 (I)

✏ 策划说：“要有车~”

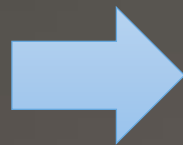
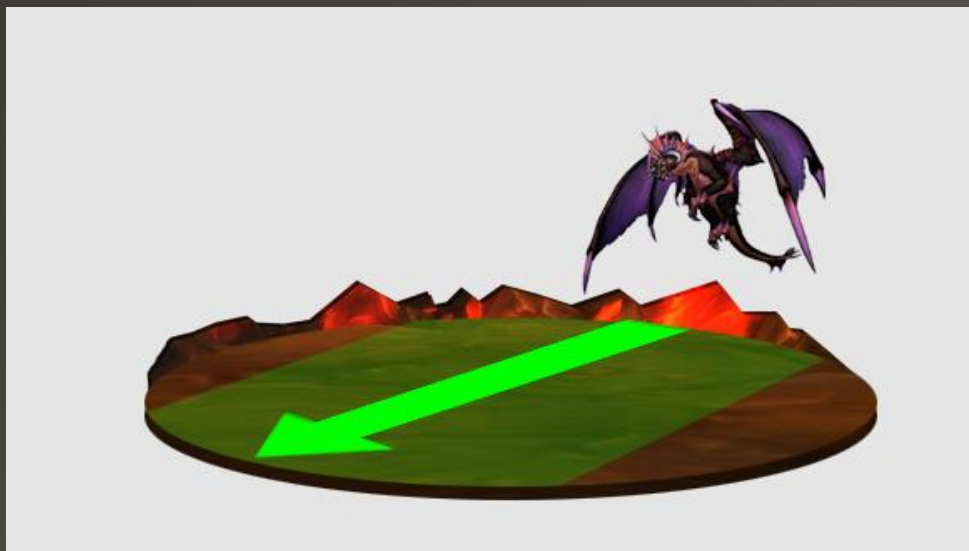
✏ 于是就有了车

有玩家最近发现，出现在《辐射 3》DLC“断钢”里的火车根本不是火车，而是一个头戴车厢形巨型头盔的女 NPC！这样一来要让它动起来就简单啦，让这个 NPC 跑起来就是。要上车当然是不可能的，我们只能依靠万能的 MOD，而且手法同样诡异：玩家戴上一个摄像机视角在别处的头盔，这样一来就好像是坐进了火车里了。

《辐射 3》用的引擎叫 Gamebryo，用它开发过的游戏有一个共同点——里头一辆能开的车都没有。仔细回想一下，《上古卷轴》系列的交通工具只有马，《辐射 3》有车但都是坏的。如何在一个不支持车辆物理效果的引擎中做出能开动的火车呢？Bethesda 用的办法简直“碉堡”。



局限性认识 (II)



万能兔子

问题解决

✎ 尺有所短 (UE3 更新)

正交性

✎ 不让“引擎不支持.....”成为万能借口 (android -> miui)

✎ 及时发现和躲开伪命题

80% 的伪命题的根源
是性能问题

技术负债

- ✎ 和公司的负债一样，并不总是坏的
- ✎ 往往是突击能力强的表现 (和技术上限的正相关性)
- ✎ 以正合，以奇胜 (hackers & scientists)

“我们选了错误的引擎，now what?”

- ✎ 跟给一个开发中的游戏换引擎比起来，给一辆车换发动机容易得就好像给你的PC换一块硬盘
- ✎ 回顾选引擎时的主要考虑，看看当时的前提是否依然有效
- ✎ （有的时候）不是引擎的问题，是人的问题

回到三个问题

Game Engine Topics

✓ 十年变迁 (2006-2016)

✓ 一次技术迭代周期

前期：引擎的评估和技术选型

后期：引擎的运用，技术改造和问题解决

 下一个十年

下一个十年

✎ 从一个数据说起



峰值速率

20G bps

用户面延时

(URLLC)

0.5ms

4G

1G bps

10 ms



Latency Numbers Every Programmer Should Know

1	Latency Comparison Numbers					
2	-----					
3	L1 cache reference	0.5	ns			
4	Branch mispredict	5	ns			
5	L2 cache reference	7	ns			
6	Mutex lock/unlock	25	ns			
7	Main memory reference	100	ns			
8	Compress 1K bytes with Zippy	3,000	ns	3	us	
9	Send 1K bytes over 1 Gbps network	10,000	ns	10	us	
10	Read 4K randomly from SSD*	150,000	ns	150	us	~1GB/sec SSD
11	Read 1 MB sequentially from memory	250,000	ns	250	us	
12	Round trip within same datacenter	500,000	ns	500	us	
13	Read 1 MB sequentially from SSD*	1,000,000	ns	1,000	us	1 ms ~1GB/sec SSD, 4X memory
14	Disk seek	10,000,000	ns	10,000	us	10 ms 20x datacenter roundtrip
15	Read 1 MB sequentially from disk	20,000,000	ns	20,000	us	20 ms 80x memory, 20X SSD
16	Send packet CA->Netherlands->CA	150,000,000	ns	150,000	us	150 ms

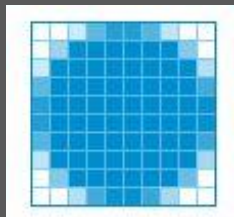
What happens when your internet connection is much much faster (10x) than your disk seek?



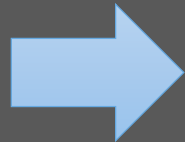
What do we see?



2160 x 1200 FOV 110

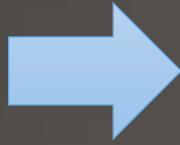


15 像素 / 度



What we got in 2016.

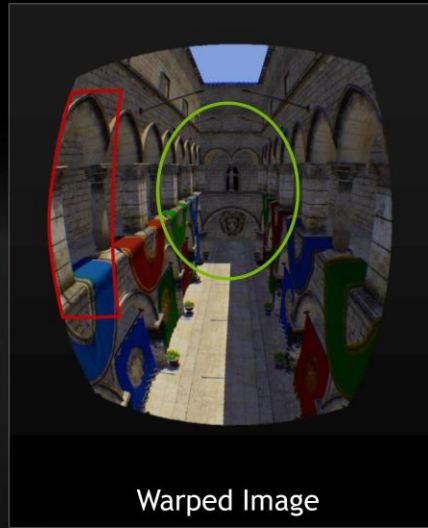
How are we “connected”?



What we got in 2016.

Multi-Res Shading

RESULT: GPU RENDERS MANY PIXELS THAT NEVER MAKE IT TO THE SCREEN

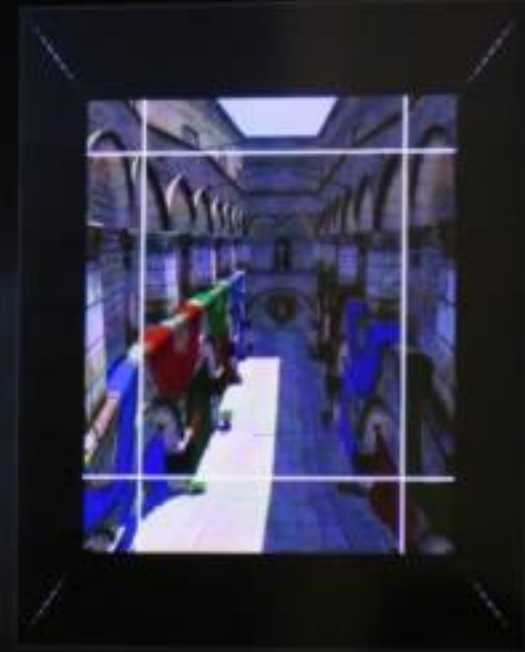


RENDER IN A SINGLE PASS!

Maxwell multi-projection accelerates geometry broadcast



Each viewport is rendered at the proper resolution for the final warped image



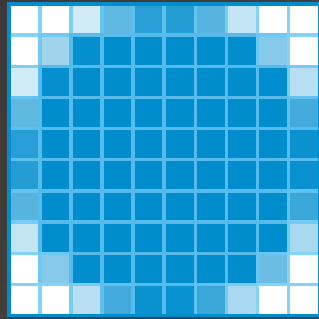
What's happening.

Foveated Rendering



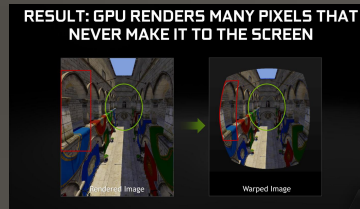
What's happening.

What will we see (soon)?

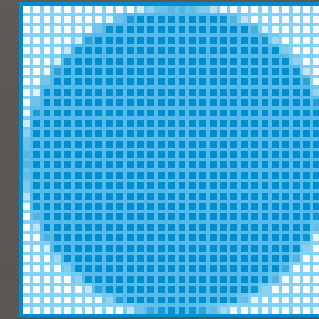


2160 x 1200

15 像素 / 度



Multi-Res Shading
Foveated Rendering



4k * 4k * 2

30 像素 / 度



What will we see (not that soon)?






Presence

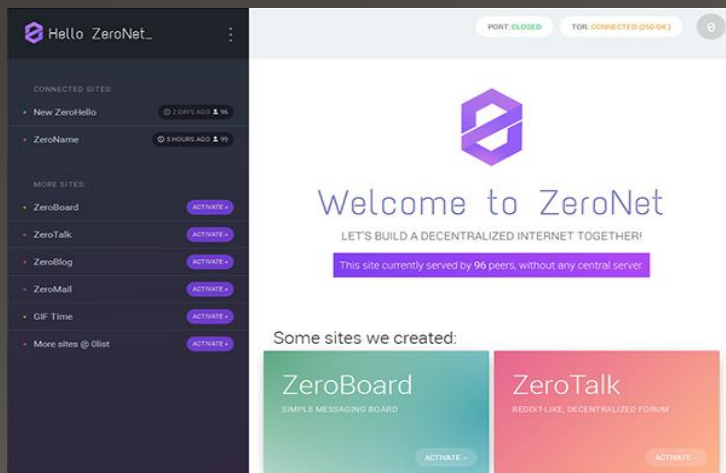
It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change.

- Charles Darwin

Get ready for it.

If we have some time left...

-  zeronet -> blockchain + bittorrent
-  WestWorld -> strong AI with machine learning
-  Google Earth VR -> «Earth» online



And even more time...



游戏A希望使用你的魔兽世界账号，授权吗？

- 获取你的公开信息（种族，职业，头像等）
- 向你的背包添加物品
- 寻找与你共同玩这个游戏的魔兽世界好友



Game as a Service

One thing to take away

It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is most adaptable to change.

- Charles Darwin

Thank You

顾露
西山居