# 章节一.物理模型的有效水模拟

本章描述了一个在 GPU 上模拟和渲染大型水体的系统。 该系统将基础网格的几何起伏与动态法线贴图的生成相结合。 该系统已被证明适用于实时游戏场景，已在 Cyan Worlds 的《Uru: Ages Beyond Myst》中广泛使用，如图 1-1 所示。



## 目标和范围

在过去的几年里，实时渲染技术已经从离线渲染世界中迁移出来。 快速傅立叶变换 (FFT) 技术，如 Tessendorf 2001 所述，为足够大的采样网格产生了令人难以置信的真实感，并且可以在消费级 PC 上实时处理中等大小的网格。 基于体素的 Navier-Stokes 方程简化形式的解决方案也是可行的（Yann 2003）。 尽管我们还没有达到尖端的离线流体模拟的地步，如 Enright 等人- 2002年，但差距正在缩小。 到本章出版时，FFT 库很可能可用于顶点和像素着色器，但在撰写本文时，即使是这些技术的实时版本也仅限于在 CPU 上实现。

与此同时，能够在 GPU 上运行的简单水模拟模型也在不断向上发展。 伊西多罗等人 （2002） 描述了在顶点着色器中对四个正弦波求和以计算表面高度和方向。 Laeuchli 2002 提出了一个使用三个 Gerstner 波计算表面高度的着色器。

我们从简单的正弦函数求和开始，然后酌情进行稍微复杂的函数。 我们还将这项技术扩展到像素着色器领域，使用周期性波函数的总和来创建动态平铺凹凸贴图，以捕捉水面的更精细细节。

本章重点解释系统参数的物理意义，表明用正弦波的总和来近似水面并不像经常提出的那样特别。 我们特别关注将我们从底层模型带到实际实现的数学； 数学是扩展实现的关键。

该系统专为从小池塘到海洋的水体设计，从海湾或岛屿上看。 虽然不是严格的物理模拟，但它确实提供了令人信服、灵活和动态的水渲染。 由于模拟完全在 GPU 上运行，因此无论是人工智能 (AI) 还是物理过程，都无需为 CPU 使用率而苦苦挣扎。 因为系统参数确实有物理基础，所以它们比通过反复试验找到它们更容易编写脚本。 使系统作为一个整体动态——除了它的分量波——增加了额外的生命水平。

## 正弦函数求和

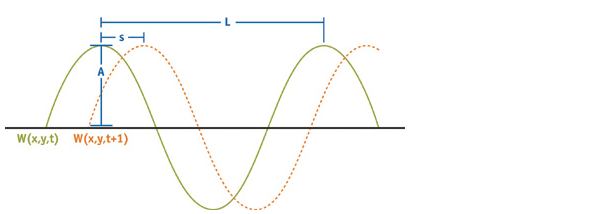
我们运行两种表面模拟：一种用于表面网格的几何波动，另一种用于该网格上法线贴图中的波纹。 两种模拟基本相同。 水面的高度由简单周期性波的总和表示。 我们从对正弦函数求和开始，然后逐渐转向更有趣的波形。

正弦函数的和给出了描述所有点的水的高度和表面方向的连续函数。 在处理顶点时，我们根据每个顶点的水平位置对该函数进行采样，使网格符合其细分到连续水面的限制。 在几何分辨率以下，我们将技术继续用于纹理空间。 我们通过简单的像素着色器操作对正弦近似和的法线进行采样渲染到渲染贴图，为表面生成法线贴图。 为每一帧渲染我们的法线贴图允许我们有限的一组正弦波独立移动，大大增强了渲染的真实感。

事实上，我们的水纹理中的细波支配了我们模拟的真实感。 我们的波浪表面的几何起伏提供了一个更微妙的框架来呈现这种纹理。 因此，我们有不同的标准来选择几何波和纹理波。

### 选择波浪

我们需要一组参数来定义每个波。 如图1-2所示，参数为：



单个Sin波函数的参数：

* 波长 (L)：世界空间中波的波峰到波峰之间的距离。 波长 L 与频率 w 相关，因为 w = 2/L。
* 振幅（A）：从水面到波峰的高度。
* 速度（S）：波峰每秒向前移动的距离。 将速度表示为相位常数很方便，其中 。
* 方向 (D)：垂直于波峰行进的波前的水平向量。

然后每个波的状态作为水平位置 (x, y) 和时间 (t) 的函数定义为：

** 1**

所有的表面相加为：

 2

所有波i相加。

为了提供场景动态的变化，我们将在约束范围内随机生成这些波参数。 随着时间的推移，我们将不断地淡出一个波，然后使用一组不同的参数将其淡入。 事实证明，这些参数是相互依赖的。 必须注意以令人信服的方式为每个波生成一整套参数。

### 法线和切线

因为我们的表面有一个明确的函数，所以我们可以直接计算任何给定点的表面方向，而不是依赖于有限差分技术。 我们的副法线 B 和切线 T 向量分别是 x 和 y 方向的偏导数。 对于 2D 水平面上的任意 (x, y)，表面上的 3D 位置 P 为：

 3

则 x 方向的偏导数为：

 4a

 4b

同样，切向量为：

 5a

 5b

法线由副法线和切线的叉积给出，如：

 6a

 6b



在放入函数 H 的部分之前，请注意公式 3-6 中的公式是多么的方便。 两个偏导数的求值给了我们切线空间的九个分量。 这是我们使用高度场来近似我们的表面的直接结果。 即 P(x, y). x = x 和 P(x, y). y = y，它们成为偏导数中的零和一。 它只对这样的高度场有效，但对我们选择的任何函数 H(x, y, t) 都是通用的。

对于 1.2.1 节中描述的高度函数，偏导数特别便于计算。 因为和的导数是导数之和：

 7

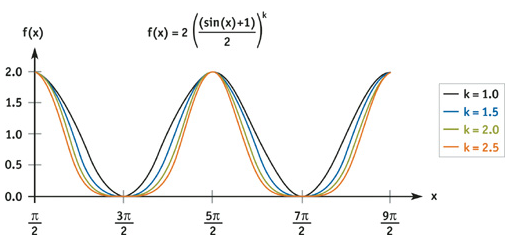
所有i相加。

对直接求和正弦波产生的波的一个常见缺点是它们波峰波谷太平，即真正的波具有更尖锐的波峰和更宽的波谷。 事实证明，正弦函数有一个简单的变体，可以完全可控地产生这种效果。 我们在确保正弦函数偏移为非负值时将其添加到指数 k。现在函数及其关于 x 的偏导数为：



 8

图 1-3 展示了受常数 k 影响的函数生成的波形。 这是我们实际用于纹理波的函数，但为简单起见，我们继续用简单的正弦和来表示波，并注意我们必须在哪些地方解释底层波形的变化。

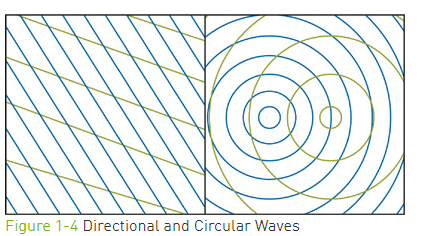


### 几何波

我们将函数限制为四个几何波。添加不涉及新概念，只涉及更多相同的顶点着色器指令和常量。

**定向的或圆形**

我们可以选择圆波或定向波，如图1-4所示。定向波需要的顶点着色器指令稍微少一些，但在其他情况下，选择取决于模拟的场景。



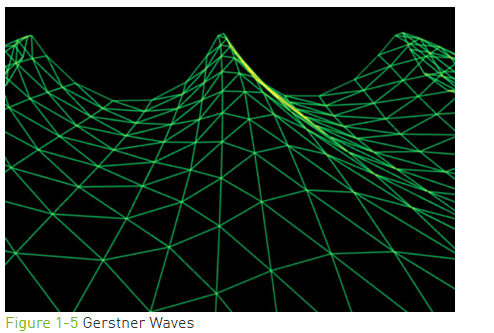
对于定向波，方程1中的每个在波的计算过程中都是常数。对于圆形波，必须在每个顶点计算方向，且方向仅为从波的中心到顶点的归一化向量：



对于大型水体，定向波通常更可取，因为它们是风浪的更好模型。对于波源不是风的较小水池（如瀑布底部），最好使用圆形波浪。圆波还有一个很好的特性，即它们的干涉图案永远不会重复。这两种波形的实现非常相似。对于定向波，波浪方向是从风向的某个方向范围中随机抽取的。对于圆形波，波中心是从某个有限范围内随机抽取的（例如瀑布撞击水面的线）。接下来的讨论集中在定向波上。

**Gerstner Waves模型**

为了进行有效的模拟，我们需要控制波浪的陡度。如前所述，正弦波的外观是圆形的——这可能正是我们想要一个平静的田园池塘所需要的。但对于波涛汹涌的大海，我们需要形成更尖锐的山峰和更宽的波谷。我们可以使用方程 8，因为它们产生了所需的形状，但我们选择了相关的 Gerstner 波。 Gerstner 波函数早在计算机图形学之前就已经开发出来，用于在物理基础上模拟海水。因此，Gerstner 波贡献了一些表面运动的微妙之处，这些细节非常令人信服，但并不明显。 （有关详细描述，请参阅 Tessendorf 2001。）我们在这里选择 Gerstner 波，因为它们具有经常被忽视的特性：它们通过将顶点移向每个波峰形成更尖锐的波峰。因为波峰是我们表面上最尖锐（即最高频率）的特征，这正是我们希望顶点集中的地方，如图 1-5 所示。



Gerstner wave 函数：

 9

这里是一个控制波浪陡度的参数。 对于单个波，会给出一个普通的正弦波，而给出一个尖锐的波峰。 应避免较大的值，因为它们会导致在波峰上方形成环路。 通常来说，我们可以将的作为艺术家制作“陡度”的参数，允许范围为 0 到 1，我们产生的波浪可以使用函数从完全平滑的波变为最尖锐的波。

请注意，函数 3 和 9 之间的唯一区别是顶点的横向移动，高度是一样的。 这意味着我们不再是严格意义上的高度函数。 也就是，但是函数还是很容易区分的，并且有一些容易消除的项。 在这里将推导过程保留，读者可以自行推导练习，我们看到切线空间基向量是：

 10

 11

 12



这些方程不像方程 4b、5b 和 6b 那样简洁，但结果证明它们的计算效率很高。

在波峰形成环的情况下，仔细观察法线的 z 分量会得到有趣的结果。 虽然 Tessendorf (2001) 从流体动力学的 Navier-Stokes 描述和“Lie Transform Technique”中获得了他的“choppiness”效应，但最终结果是在频域中表达的 Gerstner 波的变体。 在频域，可以避免和检测到波顶的循环，但在空间域，我们可以清楚地看到发生了什么。 当之和大于 1 时，我们的法线的 z 分量在峰值处会变为负值，因为我们的波会在自身上循环。 只要我们让我们的总和总是小于或等于 1，我们就会形成尖峰，不会形成环。

**波长和速度**

我们首先选择合适的波长。 与其追求真实世界的分布，我们更愿意将我们能使用的少数波的影响最大化。 相似长度波浪的超级定位突出了水面的动态。所以我们选择一个中间波长并生成一半到两倍长度之间的随机波长。 中间值波长是在代码过程中编写的，它会随着时间而变化。 例如，暴风雨期间的海浪可能比晴天和平静时大得多。 请注意，我们无法更改有效波的波长。 即便是逐渐变化，波峰也会从原点向外扩张或向着原点收缩，样子会非常不自然。 因此，我们改变了当前的平均波长，随着时间的推移，波会消失，它们会根据新的长度重新生成。方向也是如此。

给定一个波长，我们可以很容易地计算出它穿过表面的速度。忽略高阶项，水的色散关系（Tessendorf 2001）给出：

 13

其中，w是频率，g是重力常数，与我们使用的任一单位一致（例如），L是波的波峰到波峰长度。

**振幅**

如何处理振幅是一个要紧的问题。虽然可能存在振幅作为波长和当前天气条件的函数的推导的结果，但我们使用一个恒定的（或脚本化的）比率，在编码时指定。更准确地说，除了中值波长，艺术家还指定了中值振幅。对于任何大小的波，其振幅与波长之比将与中值振幅与中值波长之比相匹配。

**方向**

波的传播方向完全独立于其他参数，因此我们可以根据选择的任何标准自由选择每个波的方向。如前所述，我们从一个常数向量开始，它大致是风向。然后，我们从与风向成恒定角度的方向中随机选择。该恒定角度在可以在参数时指定，也可以编写在脚本中。

### 贴图波浪

我们计算到纹理中的波与它们的顶点表兄弟具有相同的参数化，但是有不同的限制。 首先，在纹理中，捕获广谱频率更为重要。 其次，纹理更容易形成图案，破坏了波纹的自然外观。 第三，对于给定波长，只有特定的波方向会保持整体纹理的平铺。 另外，请注意这里的所有数量都是以纹素为单位，而不是世界空间距离。

我们目前使用大约 15 个不同频率和方向的波，使用两到四个Pass。 四个Pass听起来可能有点过多，但它们是渲染在256x256 的目标纹理中，而不是在主帧缓冲区上。 在实际情况中，生成法线贴图的填充率的影响可以忽略不计。

**波长和速度**

同样，我们从选择波长开始。 我们在纹理保存的波长范围将会受到限制。 显然，如果要平铺纹理，则正弦波必须至少重复一次。 将最大波长设置为 TEXSIZE，其中 TEXSIZE 是目标纹理的尺寸。 当波长接近 4 texel 时，波将退化为锯齿图案，因此我们将最小波长限制为 4 texel。 此外，较长的波长已经通过几何波来近似，因此我们在选择时倾向于较短的波长。 我们通常选择大约 4 到 32 texel 之间的波长。 凹凸贴图每 50 尺平铺一次，32 纹素的波长对应于大约 6 尺。 这使得几何波长范围从大约 4尺向上，而纹理波长范围从大约 6 英向下，有一点重叠。

波速计算与几何形式相同。 等式8中的指数控制波峰的锐度。

**波幅和精度**

波幅的确定就像我们对几何波所做的那样，保持幅度与波长的比率恒定，即 kAmpOverLen。 这带来了一个有趣的优化。

请记住，我们这里不关心高度函数； 我们只是在构建一个法线贴图。 我们的查找纹理包含，其中 u 是从 0 到 1 的纹理坐标。我们将原始余弦值存储在查找纹理中而不是法线中，因为实际上将余弦转换为旋转法线更容易，而不是存储在法线中并尝试使用纹理旋转它们。

我们通过将查找纹理渲染到渲染目标来评估我们的正弦和的法线。 用空间表示方程 7，我们有：

 14

其中 u 和 v 在渲染贴图上从 0 变化到 1。 我们在顶点着色器中计算内部项，将结果作为 u 坐标传递给像素着色器中的纹理查找。 外部项和作为常量传入。 然后，生成的像素着色器是每波纹理查找的常数倍。 我们注意到，要使用等式 8a 中的尖峰波函数，我们只需在查找表中填写：



而不是，并传入作为外部项。

使用查找表可提供速度和灵活性。 但是，正如处理器速度与内存访问时间的相对增长速度已经使 CPU 上的查找表失宠一样，我们可以期待 GPU 上的相同演变。 展望未来，当我们选择查找纹理而不是直接算术计算时，我们希望能有更多的区别。 特别是，现下使用查找表时，我们必须重新生成查找表来改变波浪的锐度。

与我们在顶点着色器中构造几何法线的方法不同，在创建纹理波时，我们非常关注精度。 输出法线的每个分量必须表示为具有 8 位精度的有偏、有符号、定点值（Signed fixed）。 如果表面变得非常陡峭，x 或 y 分量将大于 z 分量并将Z分亮限制为 1。如果表面总是很浅，x 和 y 分量将总是接近 0 并出现量化误差。 在这里，我们更倾向于后一种情况。 如果我们可以为 x 和 y 分量的值设置严格的界限，我们可以缩放纹理中的法线以最大化可用精度，然后在使用它们时“取消缩放”它们。

检查生成的法线的 x 分量，我们首先看到余弦函数和方向向量的 x 分量都在区间 [-1，1] 上。 频率和幅度的乘积是有问题的，因为每个波的频率和幅度都不同。

用波长表示的频率方程 7，我们有：



高度主要由振幅较大的波控制，而表面方向主要由振幅与波长之比较大的波控制。

我们首先使用这个结果来证明在所有波中具有恒定的振幅与波长比是合理的，因为我们的波数量非常有限，我们选择忽略那些比率较小的波。 其次，由于这个恒定的比率，我们知道我们的波的 x 和 y 分量的绝对值被限制为小于 kAmpOverLen x 2，并且总数被限制为 numWaves x kAmpOverLen x 2。所以为了在求和过程中保持分辨率，我们写成：



并在我们使用它们时按 numWaves x 2 x kAmpOverLen 缩放。

**方向和平铺**

如果渲染目标有足够的分辨率可以在不平铺的情况下使用，我们可以将任意正弦函数累积到其中。事实上，我们可以将任意法线贴图累积到其中。例如，我们可能会随着场景中角色的移动而叠加湍流失真。或者，我们可以从一个比单个正弦波更复杂的函数开始，随着更少“波”函数的积累而获得更多的波复杂性。请记住，这种技术的力量在于波浪的相对运动。作为一个单元移动的复杂波浪模式比独立移动的简单波浪具有更少的真实性和影响力。

这些添加相对简单。然而，让渲染目标平铺，对我们积累的波函数施加了一些限制。特别要注意的是，圆形波的主要吸引力在于它们不会形成重复的图案。如果我们想让我们的纹理平铺，我们需要我们的波浪形成重复的图案，所以我们将自己限制为定向波浪。

显然，只有当纹理重复整数次时，纹理的平铺才会平铺自身。此外，对于一个给定波长的正弦波进行平铺，只有纹理的某些旋转仍然会平铺。不太明显但同样正确的是，如果我们添加的每个旋转正弦函数都会平铺，那么这些正弦函数的总和也会平铺。

因为我们通过纹理变换来旋转和缩放正弦函数，所以我们可以通过确保纹理变换的缩放旋转元素是整数来确保平铺的两个条件都满足。然后，我们通过在变换的平移中添加相位分量来平移波。请注意，因为纹理实际上是一维的，所以我们只需要关注变换后的 u 坐标。