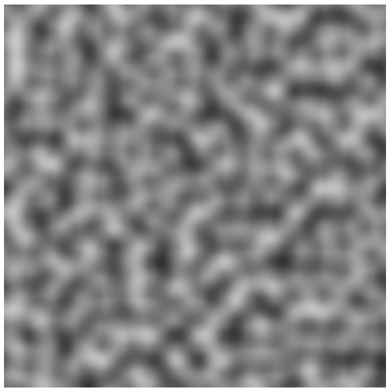
Mathematical Analysis in Game Development

Mathematical analysis is widely used in games. It can be applied to procedural generation, artificial intelligence, simulation of physical processes, 3D character modeling and other areas of game development.

Any game has a playing field where the gameplay will take place. Sometimes a developer, instead of creating the game world on his own, can entrust this task to procedural generation, when the content in the game, most often the landscape, is created using algorithms. For this, Perlin noise and its improved versions are most often used, with the addition of new methods of mathematical analysis to solve this problem.



Perlin noise (https://upload.wikimedia.org/wikipedia/commons/8/88/Perlin_noise_example.png)

For smoother values, Perlin noise uses interpolation. The most common option is linear interpolation with bezier curves. But there are other types of interpolation:

1. Cubic interpolation
2. Cosine interpolation
3. Spline interpolation:

   The curvature of any curve $y = y(x)$ is defined as

   $\kappa = \frac{y''}{(1+y'^2)^{3/2}}$, where y' and y'' the first and second derivatives of $y(x)$ with respect to x. To make the spline take a shape that minimizes the bending (under the constraint of passing through all knots), we will define both y' and y'' to be continuous everywhere, including at the knots. Each successive polynomial must have equal values (which are equal to the y-value of the corresponding datapoint), derivatives, and second derivatives at their joining knots, which is to say that

$$\begin{cases} q_i(x_i) = q_{i+1}(x_i) = y_i \\ q_i'(x_i) = q_{i+1}'(x_i) \\ q_i''(x_i) = q_{i+1}''(x_i) \end{cases} \qquad 1 \le i \le n-1.$$

This can only be achieved if polynomials of degree 3 (cubic polynomials) or higher are used. The classical approach is to use polynomials of exactly degree 3 — cubic splines.

4. RBF interpolation:

   $f(x) = \exp(x\cos(3\pi x))$, $x_k = \dfrac{k}{14}, k = 0, 1, \ldots, 14$

   $s(x) = \sum_{k=0}^{14} w_k \varphi(\|x - x_k\|)$, where φ is a radial basis function, and choose $w_k, k = 0, 1, \ldots, 14$ $w_k, k = 0, 1, \ldots, 14$ such that $s(x_k) = f(x_k), k = 0, 1, \ldots, 14$ (s interpolates f at the chosen points).

Fourier transform in procedural generation

$$\hat{f}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-ix\omega}\, dx.$$

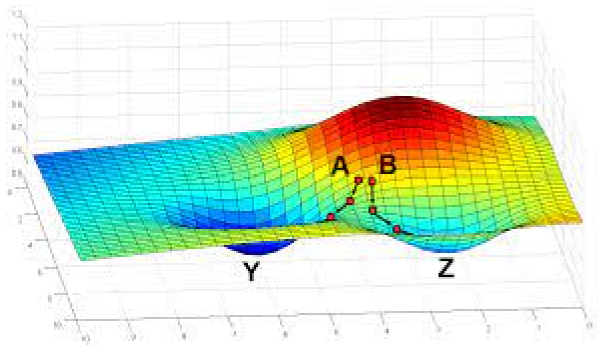
1. Creating random noise: First, an image with random noise is created. This can be white noise, which has a uniform frequency distribution, or another type of noise with specific characteristics.

2. Applying the Fourier transform: Next, the forward Fourier transform is applied to the random noise to obtain its spectral representation. This results in a set of complex numbers representing the amplitudes and phases of various frequency components.

3. Modifying the spectrum: The spectrum can be modified to control the characteristics of the texture. For example, a filter can be applied to amplify or attenuate certain frequency ranges. This allows for the creation of textures with specific structures or patterns.

4. Inverse Fourier transform: After modifying the spectrum, the inverse Fourier transform is applied to return to the image in the spatial domain. The resulting image is a procedurally generated texture with the desired characteristics.

Also mathematical analysis can be useful for creating economic strategies and RPGs. In games of this kind, the characteristics of each character can influence each other. This dependence can be described using functions. For example, damage can depend on weapons, attack speed and health. Health can be affected by regeneration rate and armor, which can reflect some of the damage. And improvements can be bought for game currency, which requires an economy and modeling of supply and demand, currency emissions, which can be described non-linearly and depend on the resources that are on the playing field. It is also important in what order to make purchases, since demand will raise supply and each purchase will increase the cost of the next. If we know the increase in value, we can calculate the integral and find out how much the item will cost, we can also take the double integral to calculate the total cost of all things that are needed. Equations can be used for the calculation of basic parameters such as time, money or territory and try to optimize them. This is where partial derivatives and gradient descent can help. Such optimizations can be useful for creating game AI that can apply these techniques to find the best game strategy.



Gradient descent (https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTU-guS0-60V8G3bG3-oT__IV2J9sr7H41keQ&usqp=CAU)

Also methods of mathematical analysis can be used to fix the game balance. Taking an already balanced character as a basis, you can choose such values of the characteristics of the new character and such coefficients that the character that was taken as the basis and the new character had a draw. An error function and its minimization can also be used, as is done in neural networks.

Sources

1) https://en.wikipedia.org/wiki/Perlin_noise
2) https://farazzshaikh.medium.com/generating-noise-using-fourier-transforms-b6ccf64afb08
3) https://www.ibm.com/topics/gradient-descent
4) https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21
5) https://en.wikipedia.org/wiki/Interpolation
6) https://habr.com/ru/articles/270465/