Precision Quantization

What is Precision Quantization

Quantization is the process of mapping values from a large set to a smaller set. This section is concerned with quantization with respect to floating point numbers. For example, a floating point may have 32 bits of data and consequently be able of representing 2^32 distinct values. In **many cases we only use a small range** of these distinct values. If we know this small range of values, we may **apply quantization and reduce our information bandwidth**.

> https://www.mathworks.com/help/comm/ug/quantization.html (https://www.mathworks.com/help/comm/ug/quantization.html)

WARNING

Precision Quantization **trades reduced precision for reduced bandwidth**.

Assume that we have a player in a large game world. Regular **floating point numbers provide multiple decimals of precision** depending on type. For a player in a large world **it is usually more than acceptable to provide precision where the |error| < 1cm**. With precision quantization we can **restrict the degree of precision and represent the data with less bits**, in this case the players position.

NOTE

Depending on game type and even object type, different levels of precision are tolerable. An intuitive example of this is that we can probably allow less precision for a car than for a player character. The small errors in precision are not noticeable for the fast car in the same way that they are noticeable for the comparatively slow character.

How to use Precision Quantization

Building on the previous example of a player in a game world it is conceivable that we may define a position vector for our player:

```
public struct PositionVector
{
    float x;
    float y;
    float z;
}
```

Assume that one unit of reference within the game world corresponds to one metric meter. It would then be **wasteful to serialize the position vector as follows** (more precision than necessary):

```
// We first define a buffer to seraialize into
PacketBuffer buffer(1024);

// We need a writer to help us with writing into the buffer
PacketWriter writer(buffer);

// Assuming that we have a position vector that we want to serialize
PositionVector position;

// We use the writer to pack each component of the vector
writer.PackFloat(position.x);
writer.PackFloat(position.y);
writer.PackFloat(position.z);
```

Since one unit of reference within the game world corresponds to one metric meter, we could decrease the precision to |error| < 1cm. Any change to a player within the world that is less than one centimeter is very unlikely to be noticeable. Therefore, it is advisable to **quantize the position vector with to reduced precision**:

```
// We first define a buffer to seraialize into
PacketBuffer buffer(1024);

// We need a writer to help us with writing into the buffer
PacketWriter writer(buffer);

// Assuming that we have a position vector that we want to serialize
PositionVector position;

// We use the writer to pack each component of the vector
writer.PackFloat(position.x, -1024, 1023, 0.01);
writer.PackFloat(position.y, -1024, 1023, 0.01);
writer.PackFloat(position.z, -1024, 1023, 0.01);
```

ℹ NOTE

It is required to **use the same precision tolerance when unpacking as when packing**.