The Pits

racing realism for the PC!

FAQ Search Memberlist Usergroups
Profile You have no new messages Logout [Jan Kohl]

Bump Mapping

newtopic postreply **The Pits Forum Index -> Tutorials**

View previous topic :: View next topic

Author	Message
Butter_ Champion Driver  Joined: 25 Feb 2003 Posts: 827 Location: State College, PA	Posted: Sat Mar 29, 2003 3:57 am Post subject: Bump Mapping quote edit delete IP I have been playing around some with the new CL program, and have a base track built. Actually I remembered more than I expected to. In about 45 min, I was able to get a whole track surface built, including the banking. Just have to add some lines through the f sections and a tsd for the s/f line to get the track part done. But, can someone please direct me toward the bumpmapping section? The section I took from the Richmond file has no bumps built in, so the entire track is a smooth as ice. Also, is it possible to just copy this part to an N2002 xml track file and compile? I want to get Jennerstown brought over to N2003, but would really like to save myself the hassle of copying every single thing over to an N2003 xml file when all I really need to change is the bump mapping and maybe a few other minor things. -Josh
Back to top	profile pm email www
GrantR Crew Chief Joined: 01 Mar 2003 Posts: 20	Posted: Thu Apr 03, 2003 4:59 pm Post subject: quote edit delete IP This might help. (Just having a minor fit of helpfulness, I guess I forgot my pills today) The way the system works is that an array of BumpMapParamsDescriptors are specified in the TrackDescriptor, where the first of these defines the default bumpmap to be used for any segment which doesn't have any custom specified bumpmaps. The each segment can, if it so chooses, do it's own thing. It does this by having a series of BumpMap sections, that function similarly to x or f or w sections, and a series of BumpMapScalar sections. A BumpMap section specifies a start dlong, and an end dlong for that segment,

followed by an index, which points to one of the BumpMapParamsDescriptors in the TrackDescriptor, which will define the bumpmap to be used along this section. If you have no BumpMap sections, then the default index 0 descriptor is used. If you have 1 section, then the bumpmap you've specified is used accross the breadth of the segment. If you have 2 or more, then the bumpmap is interpolated between sections, and outside the leftmost or rightmost bumpmap section the bumpmap is just extended out. This allows you to create custom types of bumps whereever you want. For instance at the karusell at the Nurburgring you would probably want to specify 4 bumpmap sections, a left one along the left side of the concrete that specifies the normal bumpmap to use on the asphalt to the left, then another section very close but to the right which specifies the bumpmap to use for the concrete, another section at the right side which also specifies the concrete, then another very closely to the right which specifies normal asphalt bumps for the right side of the track.

A BumpMapScalar section specifies a start dlong and end dlong, followed by a start scale and end scale. Similar interpolation rules apply between sections as to the BumpMap. Thus you can increase or decrease the size of the bumps as you see fit. For instance, making area near an apex more bumpy, or make an apron smoother, etc.

BumpMap sections and BumpMapScalar sections can't overlap themselves (of course), but they can overlap any other section type. ie: a BumpMap section can cross X, F, W and BumpMapScalar sections. a BumpMapScalar section can cross X, F, W and BumpMap sections.

I hope that makes sense.

```
TrackDescriptor
{
...
int numSegments
bool bNightTrack
double nightAmbientR
double nightAmbientG
double nightAmbientB
float northAngle
int numBumpMapParams
if (numBumpMapParams > 0)
{
int size of following data
BumpMapParamsDescriptor bumpMapParams[numBumpMapParams]
}

...
}

SegmentDescriptor
{
...
```

```
int num_X_Sections
if (num_X_Sections > 0)
{
...
}
int num_F_Sections
if (num_F_Sections > 0)
{
...
}
int num_W_Sections
if (num_W_Sections > 0)
{
...
}
enum segmentType
double headX
double headY
double headAngle // 0 to 2*Pi
double tailX
double tailY
double tailAngle // 0 to 2*Pi
bool openOnLeft
bool openOnRight
int num_BumpMaps
if (num_BumpMaps > 0)
{
int size of following data
BumpMapDescriptor bumpMaps[numBumpMaps]
}
int num_BumpMapScalars
if (num_BumpMapScalars > 0)
{
int size of following data
BumpMapScalarDescriptor bumpMapScalars[num_BumpMapScalars]
}
}


struct BumpMapParamsDescriptor
{
char name[16]; // scrambled name, not used by game
float S[4]; // scalars, high to low frequency
float D[4]; // reference dists, high to low frequency
}

struct BumpMapDescriptor
{
float headDlat;
float tailDlat;
int index; // index into bumpMapParams[] of TrackDescriptor
```

```
};

struct BumpMapScalarDescriptor
{
float headDlat;
float tailDlat;
float headScalar;
float tailScalar;
};
```

Back to top

profile pm email

GrantR
Crew Chief

Joined: 01 Mar 2003
Posts: 20

Posted: Thu Apr 03, 2003 5:05 pm Post subject: quote edit delete IP

Oh, I forgot to actually describe how the BumpMapParamsDescriptor works!

It specifies 4 pairs of numbers, which define a frequency table, which is used when computing the actual bump surface.

The scalar value of a pair specifies the size of the bump in millimeters, when at a wavelength of the distance value, in meters. wavelengths at points in between are interpolated.

Back to top

profile pm email

Butter_
Champion Driver

Joined: 25 Feb 2003
Posts: 827
Location: State College, PA

Posted: Fri Apr 11, 2003 11:22 pm Post subject: quote edit delete IP

Ok, I am NOT a program writer or code deciferer! I however have a fairly good understanding of the track editing stuff and even created a real life track for N2002. Not sure how good it was, but I haven't gotten any negative feedback anyways

But, since I don't have the skill (or programs) to break down the ptf code in a way that would best match it up to what is being said about by the gracious Grant_R, I am going to try to make some sense of this all comparing it to a whitefalcon/axaptacoder xtf file.

So, here I go, and any help/comments/suggestions would be greatly appriciated!

int numSegments is segmentcount in the xtf file

bool bNightTrack is night in the xtf file (where 1 equals night and 0 equals day)

double nightAmbientR
double nightAmbientG
double nightAmbientB are always equal to each other in the xtf file I am going to guess this is the levels of red, green, and blue that the sky is projected at. For instance, the lower the number, the "sunnier" the sky would be??? these are unk_1, unk_2 and unk_3

float northAngle is the angle (in radians?) that north is (in the sky files and thus the sun direction and glare) in comparison to the centerpoint of the track? this is unk_4

Now, the rest of this is a bit confusing to me, as in the xtf file, it is a grouping called unk_head and depending on the track, there may be a handful or two of these unk_head groupings. Now, Grant said:

```
int numBumpMapParams
if (numBumpMapParams > 0)
{
int size of following data
BumpMapParamsDescriptor bumpMapParams[numBumpMapParams]
}

...
}

SegmentDescriptor
{
...
int num_X_Sections
if (num_X_Sections > 0)
{
...
}
int num_F_Sections
if (num_F_Sections > 0)
{
...
}
int num_W_Sections
if (num_W_Sections > 0)
{
...
}
enum segmentType
double headX
double headY
double headAngle // 0 to 2*Pi
double tailX
double tailY
double tailAngle // 0 to 2*Pi
bool openOnLeft
bool openOnRight
int num_BumpMaps
if (num_BumpMaps > 0)
{
int size of following data
BumpMapDescriptor bumpMaps[numBumpMaps]
}
int num_BumpMapScalars
if (num_BumpMapScalars > 0)
{
```

int size of following data
BumpMapScalarDescriptor bumpMapScalars[num_BumpMapScalars]
}
}

struct BumpMapParamsDescriptor
{
char name[16]; // scrambled name, not used by game
float S[4]; // scalars, high to low frequency
float D[4]; // reference dists, high to low frequency
}

struct BumpMapDescriptor
{
float headDlat;
float tailDlat;
int index; // index into bumpMapParams[] of TrackDescriptor
};

struct BumpMapScalarDescriptor
{
float headDlat;
float tailDlat;
float headScalar;
float tailScalar;
};

but that is some programing stuff, that while I have a general understanding of what it means and how it is working, if someone could help me place each of the 16 fukt_n variables and the 8 unk_n variables to the variables inside for / if / then statements above and the calculations running inside of each one, it would make the exact placement of bumps a lot easier for some of us less technical folks. Also, in my opinion, the seg_end_myst_1 and seg_end_myst_2 found at the end of some segmentDescriptors should be the lines referring back to the unk_heads??

Back to top

profile pm email www

AxaptaCoder
US Pits Crew

Joined: 03 Mar 2003
Posts: 143

Posted: Fri Apr 18, 2003 6:13 pm Post subject: quote edit delete IP

Hey Guys,

I've looked at this and think I at unsterstand it, at least how the layout works

In the xtf file from Fred, near the top - under the TrackDescriptor there are several unk_head nodes. Under each unk_head is 16 fuk_t? (love the naming convention) hehe any way these map to
struct BumpMapParamsDescriptor
{
char name[16]; // scrambled name, not used by game -

So ignore these 16 - there not used anyway

the 8 unk_? after that map to the

float S[4]; // scalars, high to low frequency
float D[4]; // reference dists, high to low frequency (from the same structure)

this is then repeated for each entry in the array, so basically its a table that is global that any of the segments can use by an index - so down into the segment node past all the other sections towards the end
you see seg_end_myst_1 node that has one or more myst -
this contains three values this maps to :
struct BumpMapDescriptor
{
float headDlat;
float tailDlat;
int index; // index into bumpMapParams[] of TrackDescriptor
};

so that index maps back to the global table - i haven't looked into if this is zero or 1 based indexes, but it basically sets a the lats and the index back to the freq table.

then you see a seg_end_myst_2 then one or myst entires with 4 numbers in this entry, these are the specific segement bumps that don't map to the global but just are direct entries and maps to
struct BumpMapScalarDescriptor
{
float headDlat;
float tailDlat;
float headScalar;
float tailScalar;
};

So you can map to a global set of bumps or if a segment has a very specific bump you can get that too.

Well this is how I percieve this. Hope this helps.

Don

Back to top profile pm email

JoeWells
Driver

Joined: 08 Dec 2003
Posts: 187

Posted: Sat Feb 28, 2004 2:05 am Post subject: quote edit delete IP

Total noob question here. What exactly does bumpmapping do or effect? I hear it talked about so much, but I don't know what it means exactly.

Joe Wells

Back to top profile pm email

Butter_
Champion Driver

Joined: 25 Feb 2003
Posts: 827
Location: State College, PA

Posted: Sat Feb 28, 2004 4:57 pm Post subject: quote edit delete IP

It is what sets up the bumps in the track. With no bump mapping, you would have tracks like in N4 or N2002 that had no bumps. They were really smooth. Now, you can have a fairly smooth place like vegas, or you can have a place like darlington where the bumps are monsterous.

Back to top profile pm email www

JoeWells
Driver

Joined: 08 Dec 2003
Posts: 187

Posted: Sat Feb 28, 2004 7:34 pm Post subject: quote edit delete IP

I see. Thanks for the info. I have another question too if you don't mind.

What determines where on the track the car has grip and where it doesn't have grip?

For instance, on a track like Indianapolis, there seems to be a point in the turns where my car goes from having grip to a point where no amount of setup tweaking will matter, the car just wants to slide into the wall. Is this determined in how you create the track and or is it just the way the car physics are?

Joe Wells

Back to top profile pm email

Butter_
Champion Driver

Joined: 25 Feb 2003
Posts: 827
Location: State College, PA

Posted: Sat Feb 28, 2004 9:33 pm Post subject: quote edit delete IP

I am not sure what part of the track you are talking about, but lets say you are trying to run high in the turns. The trick with papy tracks is that the closer you run to the center (0,0) point, the faster your lap time provided you can run just as fast. (remember N3 where if you put the lefts under the line in the turns at the plate tracks you ran even faster?) IE, the low line is faster, but even more pronounced than in real life because the grip is even across the entire racing surface (save hillside). But, that grip level is optimized for the low line so that a good lap spits you out against the wall on exit. At a place like daytona, you can run that low line and never be passed because your time is faster since you are running a tighter circle around the 0,0 point and you have too much grip. At indy, the grip level is set just right for the low line, so if you move up to the high side, you are going to have to slow down more to get off of the turn without hitting the wall because you run out of room. To get the car to turn better with the setup and be able to run fast up high is not going to work because the grip level is set too low, and you will just spin the tires when you are rotating that much in the turns, so you slip and can't get the run down the straight. A small track like richmond, even with it's wide surface, a lot of people slip off of turn 4 all race long because the car has to rotate so much and has so much power. I am not so sure it is a problem with the physics though, because they seem to do the same at that track in real life. I think it is just the inertia vs turning thing that kicks the car toward the wall and papy set their grip levels so a second line is very difficult at most tracks.

Back to top

profile pm email www

JoeWells
Driver

Joined: 08 Dec 2003
Posts: 187

Posted: Sun Feb 29, 2004 1:19 am Post subject:

quote edit delete IP

Thanks for the info Butter_.

The reason I asked is that I have been redoing the AI .lp files (along with some track.ini tweaks) for all of the tracks I like to race on to make the outer line more competitive and to get more true to life looking racing from the AI and there are a couple tracks I just couldn't get to work the way I wanted, one of them being Indy (also Daytona). I tried opening up the tracks in Sandbox to see if I could figure out how Papy made the grip work, but that didn't tell me anything. Based on what you said, it seems I will have to accept the poor representation of SS racing that the Papy tracks provide because I see no way to get the higher line to be as fast as the lower line for the AI. With tracks you have to brake on, you can manipluate things a little to get what you want from the AI. With Daytona, that can't happen because you never let off the gas so there's no tricks you can pull.

Joe Wells

Back to top

profile pm email

Butter_
Champion Driver

Joined: 25 Feb 2003
Posts: 827
Location: State College, PA

Posted: Sun Feb 29, 2004 2:45 am Post subject:

quote edit delete IP

Exactly, the only way to do it is to make two grooves such as that guy did with the BR tracks by using one as a cement grip level and the other as asphalt, but you can't get the AI to "see" that so it doesn't do much good, although at the plate tracks it would work a lot better than say indy.

Back to top

profile pm email www

Display posts from previous: All Posts Oldest First Go

newtopic postreply **The Pits Forum Index -> Tutorials**

All times are GMT

Page 1 of 1

Watch this topic for replies

Jump to: Tutorials Go

You **can** post new topics in this forum
You **can** reply to topics in this forum
You **can** edit your posts in this forum
You **can** delete your posts in this forum
You **can** vote in polls in this forum
You **can** moderate this forum

Go to Administration Panel