pc world BOOKS

$13.95 (including GST)
NZ $14.90 (including GST)

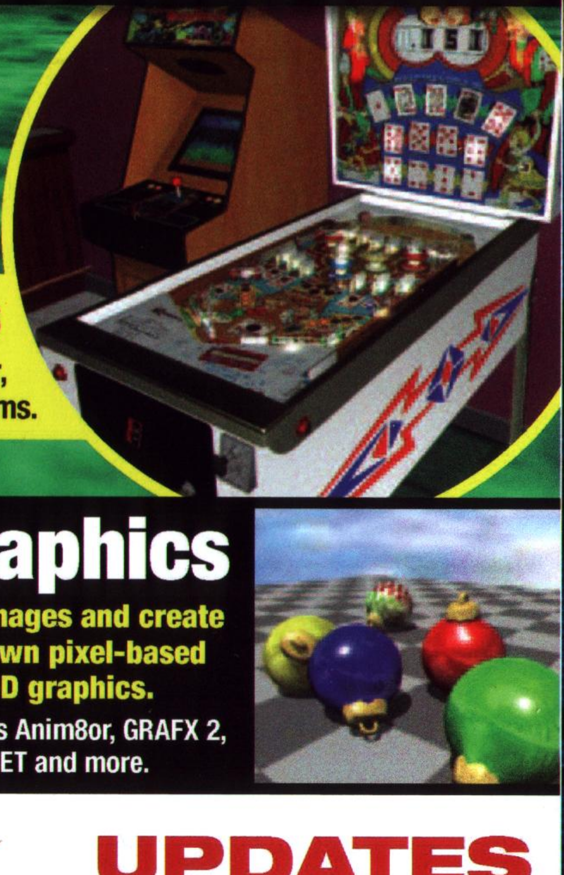
BUILD YOUR OWN GAMES

Complete beginner's guide, no programming required!

pc world BOOKS
NOT TO BE SOLD SEPARATELY

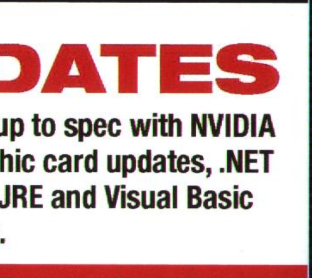
GAMES POWER TOOLS

FREE SOFTWARE



Create your own games
using Adventure Maker, Future Pinball, Game Maker, Project Dogwaffle, Reality Factory and more programs.

Graphics
Edit images and create your own pixel-based and 3-D graphics.
Includes Anim8or, GRAFX 2, Paint .NET and more.



Trial software
ONE STOP GAMES KIT!
Try out professional game development software including: DarkBASIC, FPS Creator, Pro Motion and SWiSH Max.

Java POWERED

UPDATES
Get your PC up to spec with NVIDIA and ATI graphic card updates, .NET Framework, JRE and Visual Basic runtime files.

PLUS: ALL the files required to complete step-by-step tutorials

CRAFTING A PINBALL MACHINE IN Future Pinball

www.futurepinball.com

Whether the ball bowls, bounces, shoots or slings, all ball games share the same virtue. A ball has unlimited angles it can take and you never quite know which one is next. Your opponent has the same uncertainty — which makes for plenty of upsets and reversals of fortune.

At one time, no milk bar in Australia would have been complete without a pinball machine. Admirers would gather to see the local pinball wizard rack up high scores and specials, with plenty of bells and flashing lights. The machines have become uneconomic, but very collectible — and inspiration for this excellent Australian game building software, *Future Pinball*.

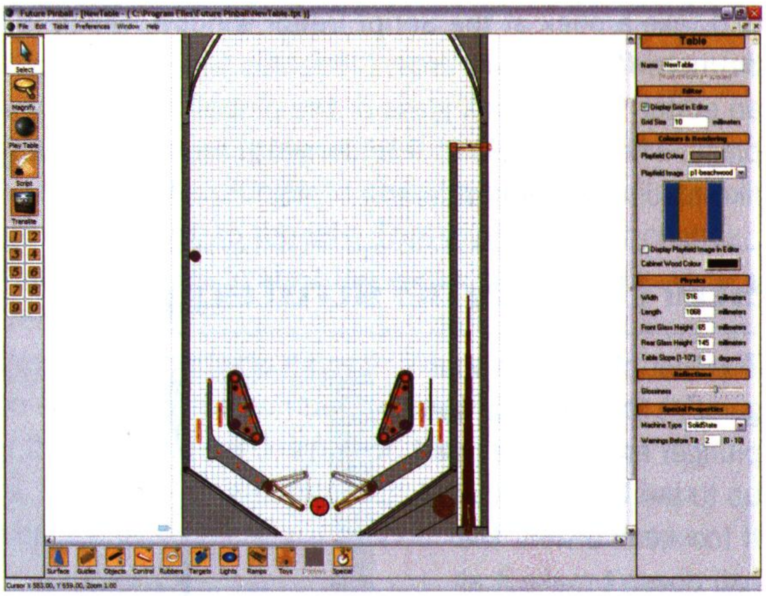
The software has two main aspects. Firstly, we have a 3-D model of the pinball machine — a virtual object with shape and physics, colour and sound. Secondly, there is the code that forms the rules of the game — winning, losing and scoring. We'll build objects and then attach code to each that will build up to a complete game. This kind of design is *object orientated* — each small part has a set of code rather than the whole game being coded "top down".

You will need a fast computer with a good display card. Locate the *FuturePinballSetup* software on the cover CD in the Pinball subdirectory. Double click to launch the standard Windows installer and follow the prompts. When the software is installed, it will ask you if you wish to run, let's agree and get going.

Getting started

At first, the authoring interface is empty. Come up to the menu and select *File - New*. After some calculation, we see a blueprint of our pinball table, with the barest minimum of objects in place (see FIGURE 1).

In fact, you can test this table right away. Over at the left, find the button marked Play Table, and click on it. The software renders a 3-D pinball machine and then "flies" to it. Press **<5>** to insert a coin. Press **<1>** to start the game. And launch a ball with the **<Enter>** key, with the flippers being the left and right **<Shift>** key. Even at this bare minimum, you have the beginnings of a game: keeping the ball from falling down the central drain. Push and hold **<Tab>** to see your score. Push **<Esc>** when you're finished (see FIGURE 2).



Future Pinball - [NewTable - (C:\Program Files\Future Pinball\NewTable.fpt)]
File Edit Table Preferences Window Help
Select
Magnify
Play Table
Script
Translite
Surface Guides Objects Control Rubbers Targets Lights Ramps Toys Special

Table
Name NewTable
Editor
Display Grid in Editor
Grid Size 10 millimeters
Colours & Rendering
Playfield Colour
Playfield Image p1-beachwood
Display Playfield Image in Editor
Cabinet Wood Colour
Physics
Width 516 millimeters
Length 1068 millimeters
Front Glass Height 65 millimeters
Rear Glass Height 145 millimeters
Table Slope (1-10°) 6 degrees
Reflections
Glossiness
Special Properties
Machine Type SolidState
Warnings Before Tilt 2 (0-10)

FIGURE 1: Future Pinball starts with a near empty blueprint of our pinball table.

Now let's look at the authoring interface in more detail. In the centre of the screen, the Blueprint shows the table from above. The programmer has placed some essential objects on the table to get us started — the flippers, launcher and so on. To the right we see the Properties Sheet for the object that's currently selected. As you click once on each object in the blueprint, this area updates with useful information about that object. Underneath the blueprint are Object Buttons that choose the kind of object that we may add.

To the left are buttons that change our mode of editing. Currently we're using Select, the most general mode where you select objects and change them. Obviously, Magnify moves closer into the blueprint. We tried Play Table earlier on. Click on Script now and take a peek at the Script Editor window, which looks a tad formidable but will be more familiar over time. Close this window for now. The Translite button opens an editor for the back glass of the pinball machine where the scoring will take place. The buttons 0 to 9 underneath are layers, a way to keep our blueprint tidy, like placing files in folders.

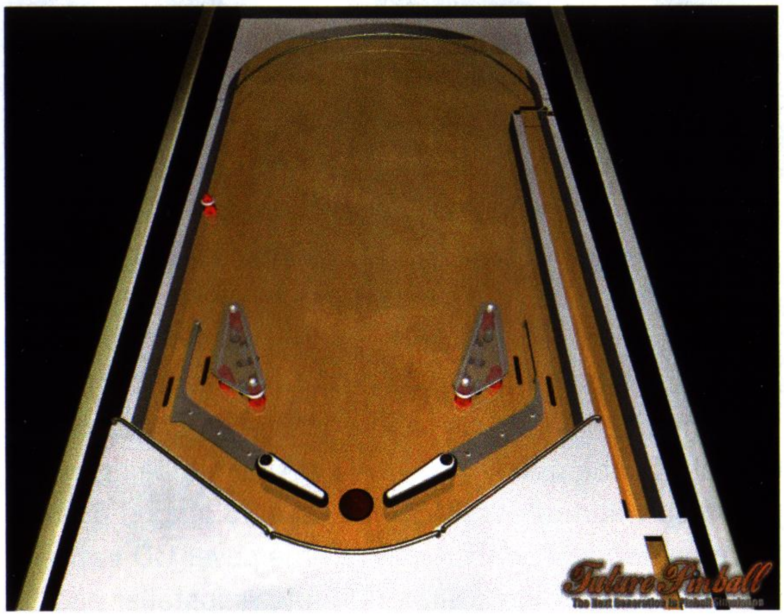


FIGURE 2: Even the empty table is a functioning 3-D simulation.

Starting game strategy

The first play in a pinball game is to pull back the plunger and fire the ball up to the top of the table. If too weak a shot, the ball doesn't make it all the way, if too hard it rolls all the way to the left and bounces uncontrollably. A "skill shot" involves plunging the ball just enough to go straight down the middle of the table. That's where we need to place a target that rewards the player for their skill.

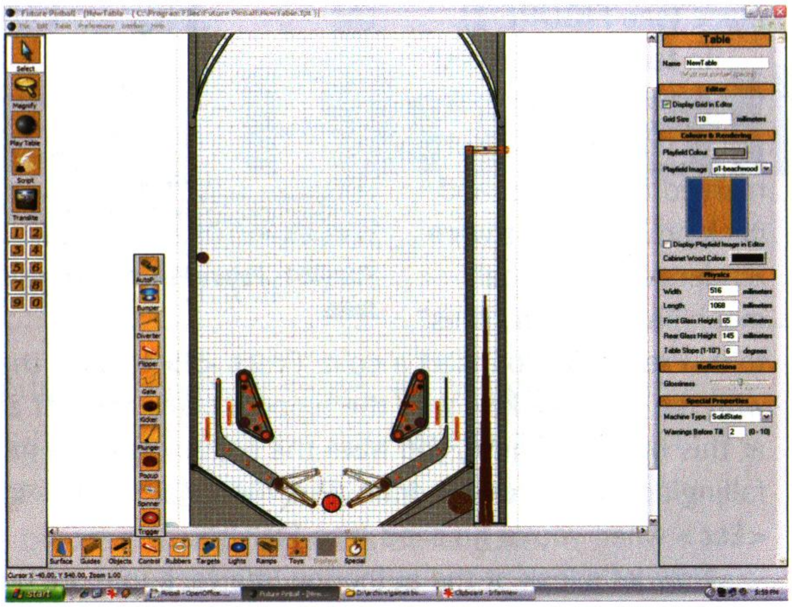


Table
Name NewTable
Editor
☑ Display Grid in Editor
Grid Size 10 millimeters
Colours & Rendering
Playfield Image p1-beachwood
☐ Display Playfield Image in Editor
Cabinet Wood Colour
Physics
Width 516 millimeters
Length 1068 millimeters
Front Glass Height 65 millimeters
Rear Glass Height 145 millimeters
Table Slope (1-10°) 6 degrees
Reflections
Glossiness
Special Properties
Machine Type SolidState
Warnings Before Tilt 2 (0 - 10)

FIGURE 3: The bumper object pops up from the menus at the bottom of the screen.

The simplest target is a pop bumper so let's try this first. Down the bottom of the interface, click the object button marked *Control* and select *Bumper* from the list that appears (see FIGURE 3).

Your mouse pointer changes to a target. Move the pointer over to the left top of the table and click to position a bumper. Place it so that the ball will hit it when the player uses less than full strength. The bumper now appears in the blueprint and the properties sheet at right will display the bumper's properties — see FIGURE 4. Most of the default settings are going to work well, but you might like to go to the last property Sound Effects and select *bumper1*.

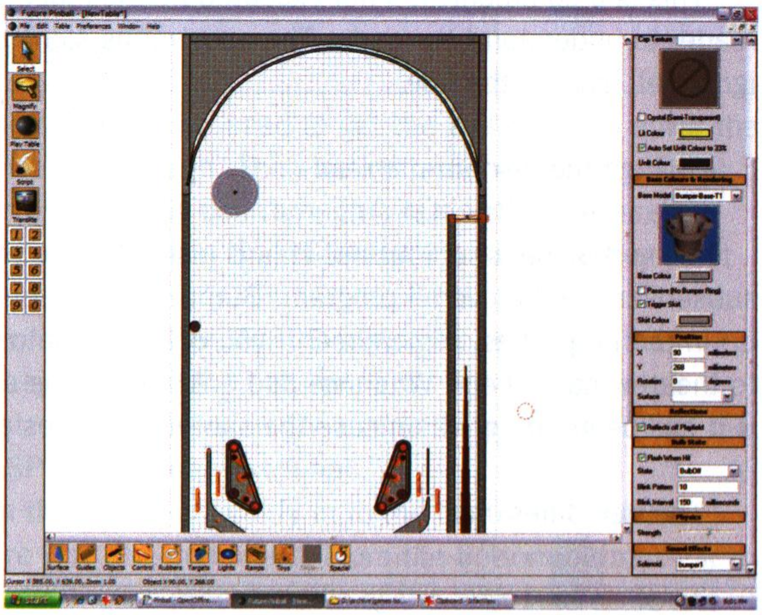


FIGURE 4: Place the bumper on the blueprint, then change the settings at the right.

By default, *Future Pinball's* plunger is a bit too athletic, so scroll the blueprint down to the bottom right, and click on the plunger to open its properties. In the Physics settings, drag Strength all the way to the left. That gives a better chance of a skill shot. But let's try it out. Hit **<F5>** to start the game, insert a coin and start the machine (5 and 1, remember?)

Can you hit the bumper? Would it be better to move it? We need to tune the game so it's not too hard, but not too easy. Move the bumper by dragging it with the left mouse button. If you can't get the bumper by a direct shot, then placing a pin might help. From the bottom buttons, click *Objects*, and pick *Peg*, by default it will be a simple pin. Place the peg in a spot where the ball will ricochet over to the bumper.

First script

When we hit the bumper, it blinks and makes a sound. But it doesn't add any score. That requires a small piece of code, which is very simple to do. First, click on the bumper and note its name at the top of the properties sheet, by default it's Bumper1. Now open the *Script Editor* and scroll to the very bottom. Here you see the code that works with the individual objects on the table. The last paragraph handles when the ball is lost over on the right-hand side. Drag your mouse over the paragraph to select it and then use **<Ctrl>-<C>** to copy it, followed by **<Ctrl>-<V>** to paste a copy after the existing text.

Change the first green line of our new paragraph to read:

```
' The Bumper has been Hit
```

This is a comment. It's not executed by the machine, rather it helps a human reader to understand the code. The apostrophe signals that the machine can ignore all else on the line.

Change the next line to read:

```
Sub Bumper1 _ Hit()
```

We're making a subroutine. This is a small piece of code or routine that is subservient to the overall program. Bumper1 is our bumper — the name that was up in the properties, remember? If you changed that name then you would refer to the new name here. When we add `_ Hit()` to the end of the name we're asking if it has been hit, in which case the subroutine takes place.

The next lines:

```
' add some points
AddScore(10)
```

remain true. We want to add some points if the bumper gets hit. We could be generous and change the 10 up to 50.

The comment:

```
' remember last trigger hit by the ball
```

remains but we will have to make the next line read:

```
set LastSwitchHit = Bumper1
```

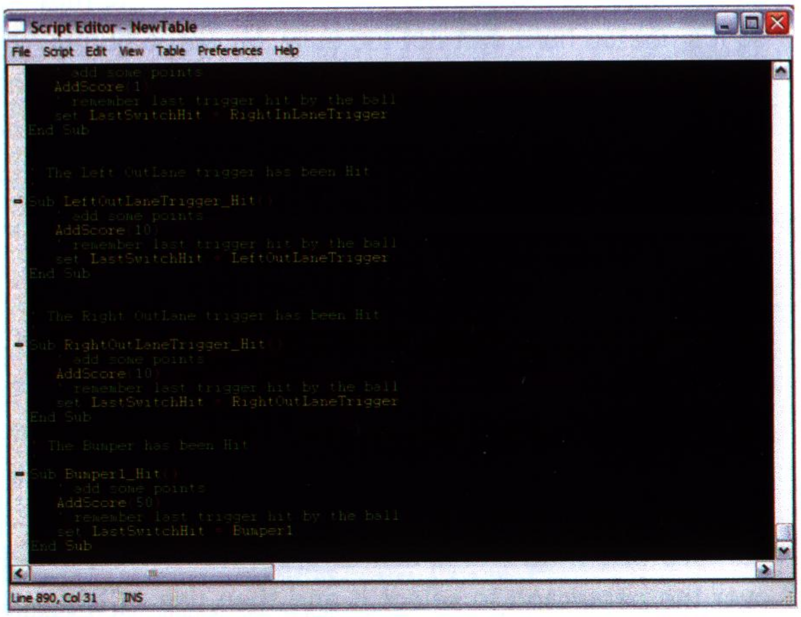
as we need to tell the main routine that the last thing that was hit was our Bumper1. Our subroutine should always tell the rest of the code any changes it has made.

Finally

```
End Sub
```

is the way we tell the computer that we've finished our subroutine. It can go do something else now (see FIGURE 5).

The important point to remember is that each object on the table has a name. It has properties we can set and check. If you copy and paste the bumper (try it, using the



```
    ' add some points
    AddScore(1)
    ' remember last trigger hit by the ball
    set LastSwitchHit = RightInLaneTrigger
End Sub


' The Left OutLane trigger has been Hit
'
Sub LeftOutLaneTrigger_Hit()
    ' add some points
    AddScore(10)
    ' remember last trigger hit by the ball
    set LastSwitchHit = LeftOutLaneTrigger
End Sub


' The Right OutLane trigger has been Hit
'
Sub RightOutLaneTrigger_Hit()
    ' add some points
    AddScore(10)
    ' remember last trigger hit by the ball
    set LastSwitchHit = RightOutLaneTrigger
End Sub

' The Bumper has been Hit
'
Sub Bumper1_Hit()
    ' add some points
    AddScore(50)
    ' remember last trigger hit by the ball
    set LastSwitchHit = Bumper1
End Sub
```

FIGURE 5: The new code is added at the bottom of the Script Editor.

Edit menu), each bumper becomes a new object that will need a subroutine. Play the game and see now that the score is raised each time you hit the bumper.

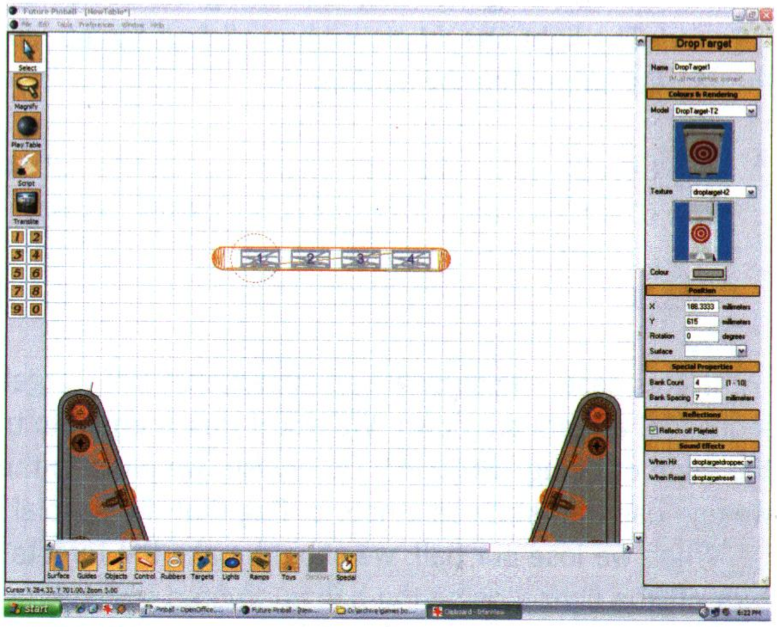
More game strategy

Whether or not they make the skill shot, we need to get the ball zooming down the table to ace them if they're not fast enough. Then we keep them down the lower end of the table unless they can hook the ball through a few well aimed angles. The best way to do that is to put a target in the middle of the table so they can't just hit the ball straight up.

We have two main levels of play to offer. For the average player, give them some targets to hit as best they can until the balls are lost. But for the better players we can set up a quest. A simple version goes something like this: "Hit the targets A-B-C-D to light the SPECIAL for an extra ball."

Let's build drop targets in the middle. Magnify in to the centre of the table. We can make four targets here with one command. From the bottom buttons click *Targets*, then *Drop*. Position your mouse in the middle of the table and click, one-drop target appears. To make more, adjust the Special Properties at right so that Bank Count is 4 and set Bank Spacing to 7mm. Position the four targets so they're at about X=180, Y=615. In the Properties Sheet, we can change their shape and decoration. From the Models drop menu, pick "DropTarget-T2". Likewise, from the Texture drop menu pick "droptarget T2". You should see old-fashioned drop targets, each with a bullseye on the front. To finish up, pick appropriate sounds for them at the bottom of the properties sheet.

For neatness, we'll make a hole for them to drop into. Select *Objects - Ornaments*, then "Orn-Hole-4bank" in the list of models. Move this hole to surround the targets. When it is in the right spot, right-click and choose *Send To Back* from the menu. The hole is behind the drop targets when you see it from the player's perspective. We need to tell the 3-D renderer that it should draw the targets in front of the hole (see FIGURE 6).



DropTarget
Name DropTarget1
Colours & Rendering
Model DropTarget-T2
Texture droptarget-t2
Colour
Position
X 188.3333 millimeters
Y 615 millimeters
Rotation 0 degrees
Surface
Special Properties
Bank Count 4 (1 - 10)
Bank Spacing 7 millimeters
Reflections
Reflects off Playfield ✓
Sound Effects
When Hit droptargetdroppec
When Reset droptargetreset

FIGURE 6: Add a group of drop targets and an ornamental hole to surround them.

These four-drop targets are a team, so we use them in a slightly more complex manner. If they all score equally, then the script is much as before:

```
' any of the drop targets got hit
Sub DropTarget1 _ Hit()
    AddScore(200)
End Sub
```

We can also interrogate one of them alone. The software knows which one was last hit and uses a variable called fpEventID to store its number. Suppose each target was to be worth more from left to right. We can script so that each case is covered one after the other.

```
' one of the drop targets got hit
Sub DropTarget1 _ Hit()
' pick one of four possible values
    Select Case (fpEventID)
            Case 1: AddScore(50)
            Case 2: AddScore(100)
            Case 3: AddScore(150)
            Case 4: AddScore(200)
    End Select
End Sub
```

When all drop targets are down, we should give the player the chance to get a special, which for our convenience they can collect from the right out lane. We should now add a light to that lane using the button at the bottom marked *Lights - Round*. The light should glow when all four targets go down. We can test that in the targets subroutine by adding:

```
' player got all of them so let's glow
If (DropTarget1.Dropped = TRUE) Then
    Light1.State = BulbOn
End If
```

In this case, the software considers that DropTarget1 is down when all the individual targets are down. We ask if this is equal to true. If so, we give Light1 the state of being "BulbOn". Otherwise, we jump to the statement End If. Later we can use that state as a note or flag that the special is there for the taking.

When we lose our ball, we should make the targets all pop back up again and turn off the light. Losing the ball is covered in a subroutine called EndOfBall. We can find that in the scripting window by using Find, or the key command <Ctrl>-<F>. There are already a lot of things that happen in this subroutine that we should not change, so we shall come down to the end of the routine, just before the End Sub command and add:

```
DropTarget1.SolenoidPulse()
Light1.State = BulbOff
```

A solenoid being the little motor that kicks the drop targets back up again. We "pulse" it so it kicks for a short period. BulbOff turns off the light.

One more bit of scripting required. If the light is on, then if they get the right lane, they should get their extra ball. In the subroutine for the right line, add:

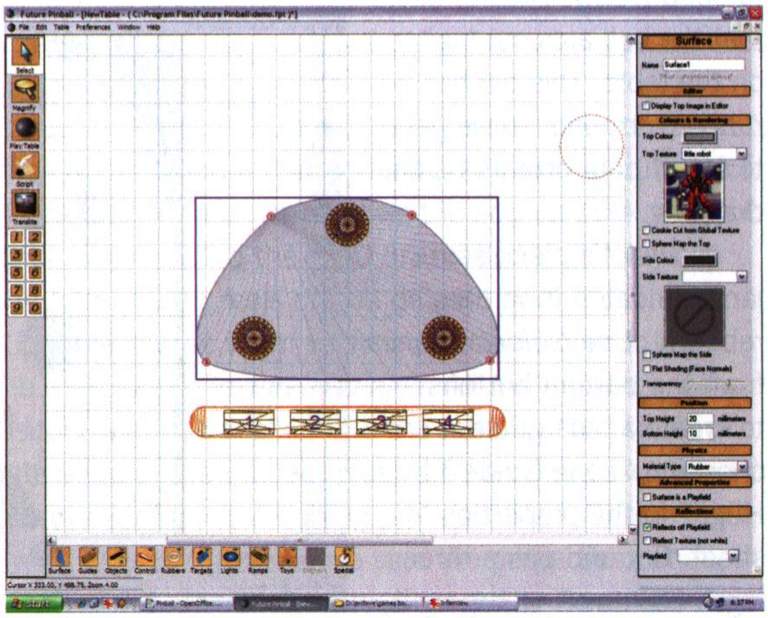
```
If (Light1.State = BulbOn) Then
     ExtraBallsAwards(CurrentPlayer) = ExtraBallsAwards(Cu
rrentPlayer)+ 1
     PlaySound "knocker"
     ShootAgainLight.State = BulbOn
End If
```

Designing the flow of the table

So that the targets only get hit from below, we'll build a protective surface above them. Add a peg and choose a model of a plastic one. Put it just above the left of the targets. Now right-click on it, and choose *Copy*. Move to the right and right-click to choose *Paste At*. Paste another peg above the right of the targets. Add a third to create a triangle. Now add a Surface, which will start as a rectangle with four control points, one at each corner. Right-click directly on each control point to select it, then select smooth. Drag the control points so that the surface surrounds the three pins, in a soft wedge shape (see FIGURE 7).

Surfaces by default will sit above the height of the ball. Lower this one so that it blocks the ball by entering the values Top Height 20 and Bottom Height 10mm.

Now we should play the game and see how it feels. One thing that feels wrong is that the ball goes down



Name: Surface1
Display Top Image in Editor
Top Texture: little robot
Cookie Cut from Global Texture
Sphere Map the Top
Sphere Map the Side
Flat Shading (Face Normals)
Top Height: 20 millimeters
Bottom Height: 10 millimeters
Material Type: Rubber
Surface is a Playfield
[✓ Reflects off Playfield]
Reflect Texture (not white)

FIGURE 7: Control points bend the shape of the wedge to make a guard.

the left and right outlets a bit too often. We should tune this. Place a plastic pin just above each of the outlets, flush with the walls, so that the ball has to arrive from the side rather than from above. By adding a circular piece of rubber on top of each pin, they become bouncy and add to the excitement.

Up top, the bumper should be arranged more carefully. If you had added a pin to help hit the bumper, remove it now. Drag the bumper to just above the triangular shape in the middle (at about X=230, Y=470). Change the cap model to "Bumper Cap T2", and Cap Texture to "Bumper Cap T2". Right-click it and use *Copy* from the pop up menu. Right-click and paste another bumper over to the left (at X=150, Y=380) and then another over to the right (at X=310, Y=380) so we have three bumpers — see FIGURE 8. If we set them up right, the ball will ricochet between them. Make sure you duplicate the script for the first bumper to handle the two new copies, changing the name of each.

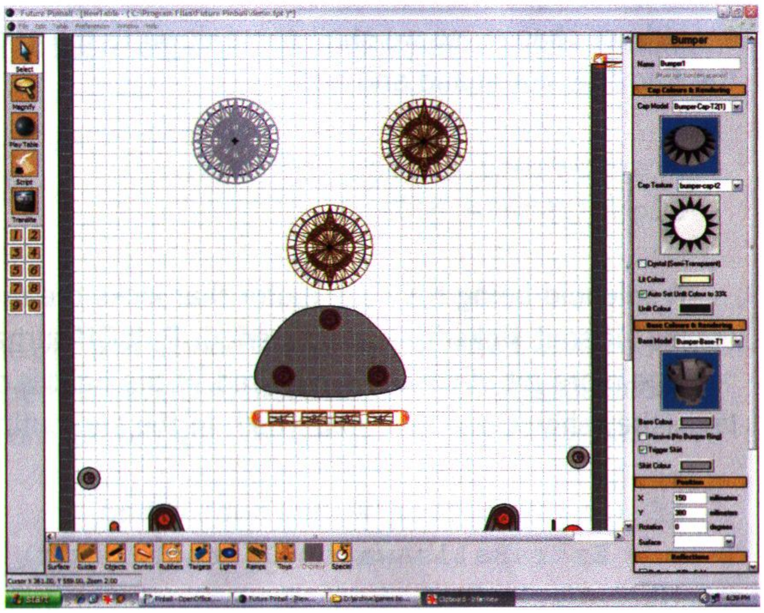


FIGURE 8: Move the bumper and duplicate it to make a set of three. Add guard pins at left and right.

Right now, the ball will first fly around to the pin at the top left and bounce back to no useful place. Move the pin up higher, to around the end of the curved guide rail (around X=33, Y=232) so that the ball arcs out higher on the table. This pin has a rubber surround, so we need to select both objects before dragging.

Now, let's place some lanes to catch it on the rebound. Use *Guides - Lanes* from the bottom buttons to create a lane guide, choosing "Lane Guide-T1-Huge" in the properties sheet. Cut and paste three more copies off to the right, each about three and a

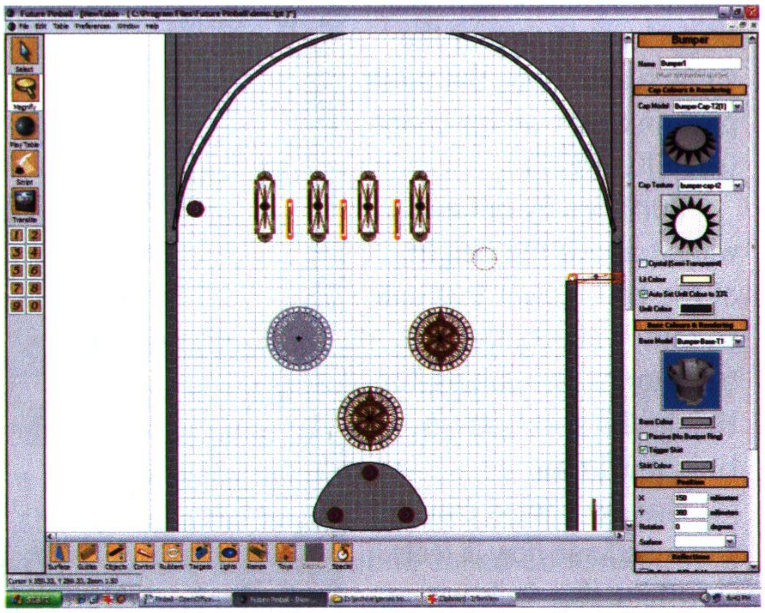


FIGURE 9: Add four lanes up top so that we have a target for skill shots.

half grid squares apart. These should be arranged slightly over to the left where the ball will fly, and above the bumpers — see FIGURE 9. Try running the game to see how the rebound works. In between each pair, place a *Control - Trigger* and choose "Trigger Wire T1" from the properties drop down. Each wire should sit so that the ball rolls over it once it enters a lane.

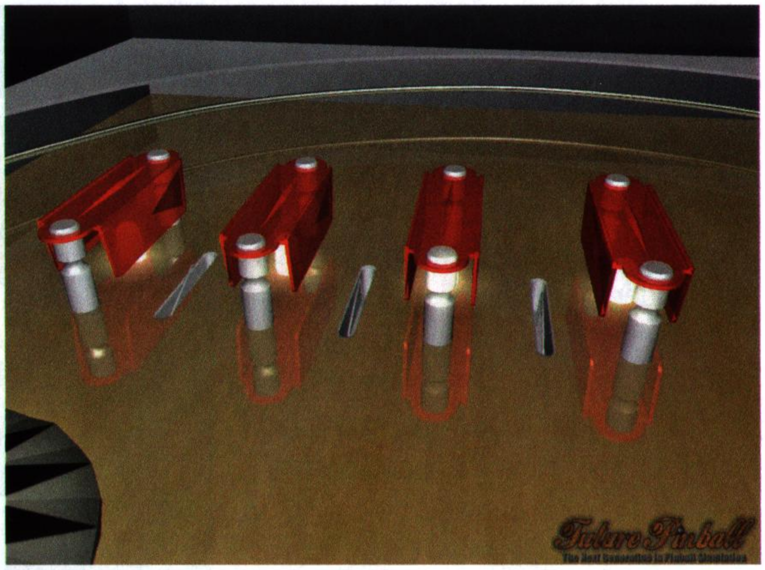


FIGURE 10: Add some bulbs and they now look quite convincing.

Each trigger wire should have a script attached that adds some score. We could also have them change the scoring of another object. Can you think of a way to: turn on a light, then have another object add more score if that light is on? Have a look at the code we have used before.

We really should make it look like a real pinball machine. Add a hole for each wire using *Objects - Ornaments* and choosing "Orn-Hole-Trigger-Long" from the list of models. Line up the hole around the wire. Also, each lane guide should have a bulb inside. Add *Lights - Bulb* and then choose "Bulb Wedge Small" from the model list. These are usually white, and should sit inside the guide. Use cut and paste to duplicate them. The lane guide in turn should be translucent, so for each, tick the box marked "Crystal (Semi Transparent)". Add a metal pin to the start and end of each lane divider, again using the cut and paste technique (see FIGURE 10).

Adding the Translite image

We should now give our machine a name and a theme. First, click on the button for the Translite over to the left. Our blueprint area now shows the square back glass of the pinball machine with a region above called the HUD or Heads Up Display. On the back glass, you can see the four scoring reels on the default machine. These could be moved and changed but we'll leave them be for now. Come over to the top menu *Table*, and select the menu item *Texture Manager*. All the images used in the table are held in this repository which is packed into the table's save file. Click on the *Import* button and locate a file in the Pinball folder on the cover disc called *Robot.bmp.* You should see a preview of a robot hurling pinballs. Click *OK*.

Now we will place the robot on the back glass. Make sure the properties panel at right reads Translite. Change the Translite Colour to white. Change the Translite Image to Robot. Tick the box Display Translite In Editor. Our robot is visible in the blueprint.

From the bottom buttons, select *Lights*, then *Bulb*. We are going to make his pinballs blink. Click on one of the pinballs shown in the picture of the robot, and position the bulb so that it's on the highlight of the ball. Untick the Render Model box — we don't want the bulb to be seen. Change the Lit Colour to white. Change the Bulb State to BulbBlink. Now using the right mouse button, click Copy and then Paste At to make copies of the bulb on each of the four pinballs. Change the Blink Interval on each so that they are out of synchronisation. Now play the game and use the **<Tab>** key to see the cool back glass effect (see FIGURE 11).

We don't get to see our robot enough. Go back to the Texture Manager. Import the file called *little robot. tga.* This robot has an alpha channel which means he has a transparent background. Click *OK*. Select the surface above the drop targets and set the Top Texture to be "little robot". Now he appears on the surface, but we see through the areas where the picture is clear (see FIGURE 12).

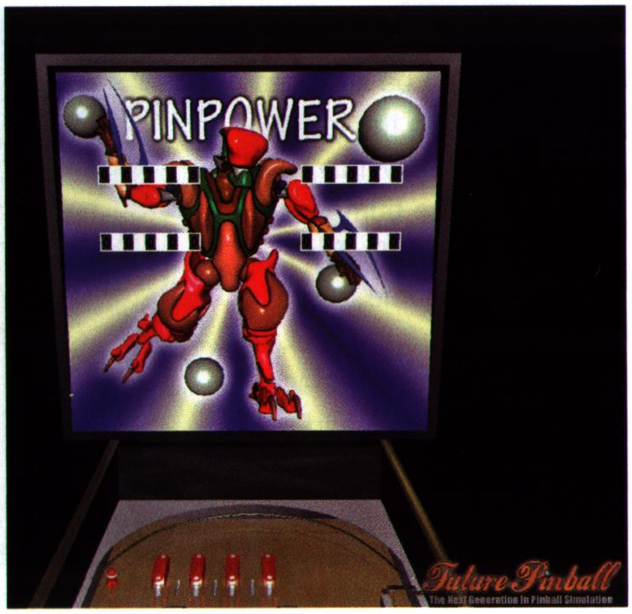
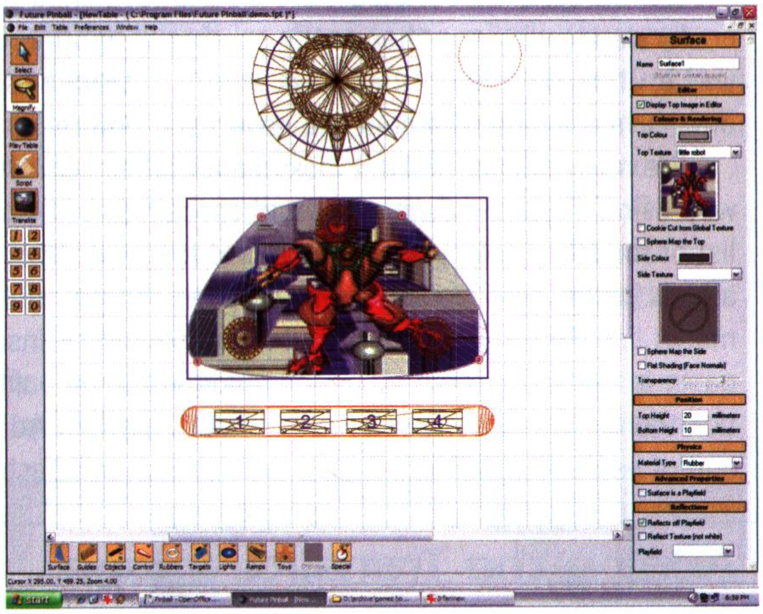


FIGURE 11: The robot image stretches over the translite.



Name: Surface1
☑ Display Top Image in Editor
Top Colour
Top Texture: little robot
☐ Cookie Cut from Global Texture
☐ Sphere Map the Top
Side Colour
Side Texture
☐ Sphere Map the Side
☐ Flat Shading (Face Normals)
Transparency
Top Height: 20 millimeters
Bottom Height: 10 millimeters
Material Type: Rubber
☐ Surface is a Playfield
☑ Reflects off Playfield
☐ Reflect Texture (not white)
Playfield

FIGURE 12: Map the little robot image to the guard wedge. This TGA image has a transparent background.

Decorating the table surface

At the moment the whole table surface is a plain wood colour — it would be nice to add some decoration. But we would like to keep the decoration in line with our play objects, so we need to export the blueprint. Go up to the menu and select *File - Generate Blueprint...* and use the dimensions as suggested. *Future Pinball* prefers all images to be sized in powers of 2.

Open it up in a paint program such as *PhotoShop*. On old pinball machines, most of the surface was lacquered wood and the decorations were painted over that. Go to the Texture Manager in *Future Pinball* and find the texture p1-Beachwood. Click on the *Export* button and save this texture next to the blueprint file. Open it up in the paint software and add it as a layer to the blueprint image. Enlarge the Beachwood texture to fill the blueprint, then arrange these layers so that the blueprint layer multiplies or darkens the wood underneath. Now you can add more layers which decorate the various objects on the table, using the transparent blueprint layer as a guide.

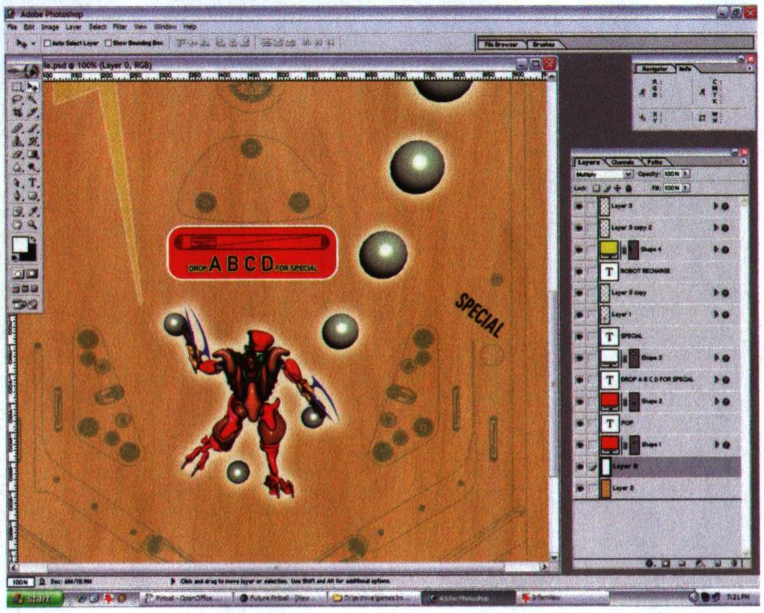


FIGURE 13: In your paint software, layer up the decals over the wooden texture using the exported blueprint as a guide.

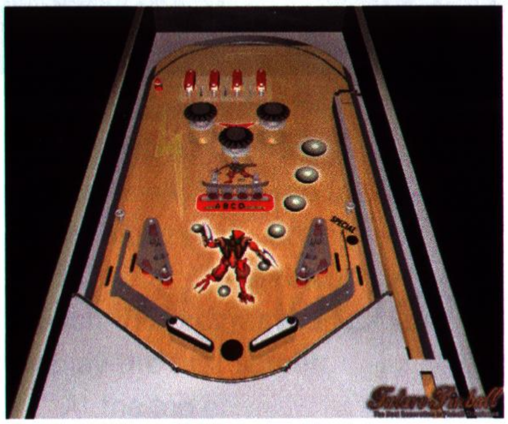


FIGURE 14: It should now line up nicely on the 3-D surface of the table.

Don't forget to place some text that tells the player the rules of each object, for example, the light on the right out lane means a SPECIAL (see FIGURE 13). A good pinball font is Impact.

Once the image looks good, hide the layer that holds the blueprint. Save the file as a layered PhotoShop image and as a flattened BMP. Import the latter into *Future Pinball*. In the table's properties, select the Playfield Image to be your imported image. Then tick "Display Playfield Image In Editor". This should line up with your objects. Play the table to test it.

We are just about ready to sign our name. Open up the *Table Info...* from the menu and fill in the details. Now the table will start up with our name (see FIGURE 14).

Although we have a perfectly functioning pinball game, there are plenty of tricks we can add. The help file will list more objects that you can add.

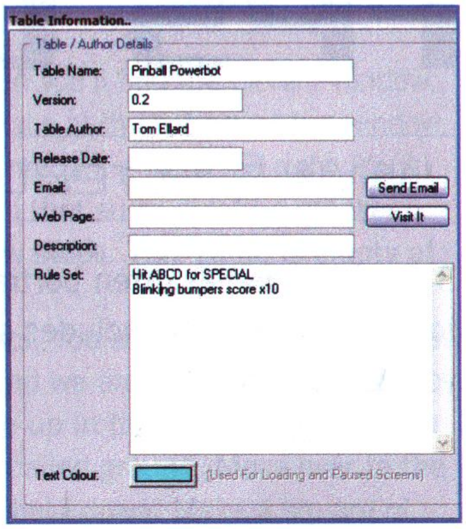


Table Information..

Table / Author Details

Table Name:	Pinball Powerbot
Version:	0.2
Table Author:	Tom Ellard
Release Date:	
Email:	
Web Page:	
Description:	
Rule Set:	Hit ABCD for SPECIAL Blinking bumpers score x10
Text Colour:	(Used For Loading and Paused Screens)

FIGURE 15: Now name your new creation.

Future Pinball

DOWNLOAD

A real time Pinball Development System that allows you to design and play your very own pinball simulation in True real time 3-D. It uses Advanced Physics to provide the best possible simulation of a real pinball machine.

File Details

File name:	FuturePinballSetup_v1.5.20060416n.exe
File size:	16.9MB
Operating system:	Windows XP
Licence:	Free
Vendor/author:	Christopher Leathley
Web site:	Visit Web site

Previous page