فصل هفدهم

انتخاب اشیاء

مقدمه :

بعضی از برنامه ها صرفا نمایش دهنده تصاویر ۲ یا ۳ بعدی ثابتی هستند و بعضی دیگر به کاربر اجازه می دهند تا اشیاء را انتخاب کرده ، حرکت دهد و غیره . انتخاب (Selection) توسط برنامه نویس بکار گرفته می شود تا معین کند ، چه شیءایی در ناحیه معینی از پنجره واقع شده است . OpenGL این امکان را فراهم می سازد که با کلیک کرسر ماوس بر روی اشیاء نام شیء انتخاب شده برگردانده شود ؛ به این پروسه انتخاب گویند . انتخاب یکی از حالات OpenGL می باشد . Feedback نیز حالتی دیگر است . در حالت Feedback از سخت افزار گرافیکی و OpenGL برای ترسیمات کمک گرفته می شود . برای مثال ترسیم اشیاء توسط پلاتر .

الگوریتم انتخاب اشیاء در OpenGL :

خلاصه الگوریتم : دریافت مختصات x,y کرسر ماوس و بررسی اینکه آیا هر شیءایی در ناحیه ای که از دنیای ۳ بعدی بصورت ۲ بعدی تصویر شده است قرار دارد یا خیر. بخاطر داشته باشید که قبل از ترسیم شیء باید یک ID به آن نسبت داد .

۱- هنگامیکه مختصات را یافتیم ، آرایه ای را برپا می کنیم تا داده های انتخاب را در خود نگه دارد :

```
glSelectBuffer(SIZE, ARRAY);
```

۲- سپس به مختصات درگاه دید نیاز داریم :

```
glGetIntegerv(GL_VIEWPORT, viewportArray);
```

۳- در ادامه باید در حالت Projection قرار گرفت تا در فضای ۲ و ۳ بعدی بتوان کار کرد . پس از این از حالت رندر خارج شده و به حالت انتخاب وارد می شویم :

```
glRenderMode(GL_SELECT);
```

۴- سپس باید به OpenGL گفت که در چه ناحیه ای باید بدنبال شیء بگردد . این کار توسط دستور زیر انجام می شود .

```
gluPickMatrix(xPos, yPos, width, height, viewport)
```

۵- سپس با فراخوانی gluPerspective برای خلق ماتریس Projection صحیح برای آن ناحیه اقدام می کنیم . سپس نیاز است تا ماتریس modelview انتخاب شود تا بتوان اشیاء را در بافر انتخاب ترسیم کرد .

قبل از استفاده از دستور gluPickMatrix ماتریس Projection جاری را ذخیره کنید . سپس از glLoadIdentity برای خلق حجم دید واحد کمک گیرید . دستور gluPickMatrix حجم دید را به موقعیت صحیح منتقل می کند . سپس تصویر گری پرسپکتیو را اعمال نمایید . صحنه دوباره ترسیم می گردد . ماتریس Projection را دوباره بازیابی کنید و ماتریس فعال را به پشت ماتریس Modelview انتقال دهید .

۶- با استفاده از glInitNames پشته نام ها را می توان پاک کرد . قبل از رندر کردن یک شیء ، باید یک ID به آن نسبت داد :

```
glLoadName(ID);
```

توسط دستور glEnd() ، ساخت یک شیء نام دار به پایان می رسد .

به اشیاء یک عدد صحیح نسبت داده می شود و لیست نام ها درپشته نام ها نگهداری می گردد . هنگامیکه انتخاب صورت می گیرد ، تمام اسامی پشته نام ها به درون بافر انتخاب کپی می شوند زیرا یک انتخاب ممکن است بیش از یک نام را برگرداند .

۷- متعاقبا باید به حالت نرمال رندرسازی بازگشت و تعداد اشیاء انتخاب شده را دانست :

```
numberOfObjectsHit := glRenderMode(GL_RENDER);
```

۸- ماتریس Projection جاری را با بازگشت به GL_PROJECTION می بندیم و سپس به ماتریس modelview خود می رویم .

۹- عاقبت ID های اشیاء کلیک شده و ذخیره شده در selectBuffer را خواهیم داشت . هر آرایه شیء از ۴ قسمت تشکیل شده است . ما فقط به دومین و چهارمین آنها نیاز داریم ، زیرا آنها حداقل عمق و شماره هویت یک شیء را ذخیره می کنند. اولین اندیس آرایه حاوی تعداد نام ها ، بر روی پشته نام ها است ، هنگامیکه انتخاب رخ می دهد . دو اندیس بعدی همانطور که ذکر شد ، حاوی حداقل و حداکثر مختصات Z تمام رئوس موجود در حجم دید می باشند .

چند نکته :

۱- glPushName() , glPopName() می توانند بجای glLoadName() , glEnd() بکار روند ولی بر روی کارت های شتاب دهنده گرافیکی مانند *Voodoo* کار نمی کنند .

۲- اگر از glLoadName قبل از رندر کردن شیءایی استفاده نکنید ID آن شیء صفر خواهد بود .

۳- اگر از quadrics استفاده می کنید ، از یک glBegin()/glEnd() خالی بین glLoadName()/glEnd() استفاده کنید ، تا بعضی از کارت های شتاب دهنده گرافیکی را دچار مشکل نکنید !

مروری بر توابع :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glGetFloatv(GLenum pname, GLfloat * params);	Procedure glGetFloatv(pname: GLenum; params: PGLfloat); stdcall; external 'OPENGL32.DLL';

توضیح :

این توابع مقدار یا مقادیر پارامتر انتخاب شده را بر می گردانند .

Pname : پارامتری است که باید توسط آرگومان Params برگردانده شود . مقادیر سمبولیک قابل قبول برای آن در جداول زیر آورده شده اند.

Params : مقدار یا مقادیر برگشتی پارامتر معین شده است .

Table 1 : State Variables for Current Values and Associated Data				
State Variable	Description	Attribute Group	Initial Value	Get Command
GL_CURRENT_COLOR	Current color	Current	1, 1, 1, 1	glGetIntegerv(), glGetFloatv()
GL_CURRENT_INDEX	Current color index	Current	1	glGetIntegerv(), glGetFloatv()
GL_CURRENT_TEXTURE_COORDS	Current texture coordinates	Current	0, 0, 0, 1	glGetFloatv()
GL_CURRENT_NORMAL	Current normal	Current	0, 0, 1	glGetFloatv()
GL_CURRENT_RASTER_POSITION	Current raster position	Current	0, 0, 0, 1	glGetFloatv()
GL_CURRENT_RASTER_DISTANCE	Current raster distance	Current	0	glGetFloatv()
GL_CURRENT_RASTER_COLOR	Color associated with raster position	Current	1, 1, 1, 1	glGetIntegerv(), glGetFloatv()
GL_CURRENT_RASTER_INDEX	Color index associated with raster position	Current	1	glGetIntegerv(), glGetFloatv()
GL_CURRENT_RASTER_TEXTURE_COORDS	Texture coordinates associated with raster position	Current	0, 0, 0, 1	glGetFloatv()
GL_CURRENT_RASTER_POSITION_VALID	Raster position valid bit	Current	GL_TRUE	glGetBooleanv()
GL_EDGE_FLAG	Edge flag	Current	GL_TRUE	glGetBooleanv()

Table 2 : (continued) Vertex Array State Variables				
State Variable	Description	Attribute Group	Initial Value	Get Command
GL_VERTEX_ARRAY	Vertex array enable	vertex-array	GL_FALSE	glIsEnabled()

GL_VERTEX_ARRAY_SIZE	Coordinates per vertex	vertex-array	4	glGetIntegerv()
GL_VERTEX_ARRAY_TYPE	Type of vertex coordinates	vertex-array	GL_FLOAT	glGetIntegerv()
GL_VERTEX_ARRAY_STRIDE	Stride between vertices	vertex-array	0	glGetIntegerv()
GL_VERTEX_ARRAY_POINTER	Pointer to the vertex array	vertex-array	NULL	glGetPointerv()
GL_NORMAL_ARRAY	Normal array enable	vertex-array	GL_FALSE	glIsEnabled()
GL_NORMAL_ARRAY_TYPE	Type of normal coordinates	vertex-array	GL_FLOAT	glGetIntegerv()
GL_NORMAL_ARRAY_STRIDE	Stride between normals	vertex-array	0	glGetIntegerv()
GL_NORMAL_ARRAY_POINTER	Pointer to the normal array	vertex-array	NULL	glGetPointerv()
GL_COLOR_ARRAY	RGBA color array enable	vertex-array	GL_FALSE	glIsEnabled()
GL_COLOR_ARRAY_SIZE	Colors per vertex	vertex-array	4	glGetIntegerv()
GL_COLOR_ARRAY_TYPE	Type of color components	vertex-array	GL_FLOAT	glGetIntegerv()
GL_COLOR_ARRAY_STRIDE	Stride between colors	vertex-array	0	glGetIntegerv()
GL_COLOR_ARRAY_POINTER	Pointer to the color array	vertex-array	NULL	glGetPointerv()
GL_INDEX_ARRAY	Color-index array enable	vertex-array	GL_FALSE	glIsEnabled()
GL_INDEX_ARRAY_TYPE	Type of color indices	vertex-array	GL_FLOAT	glGetIntegerv()
GL_INDEX_ARRAY_STRIDE	Stride between color indices	vertex-array	0	glGetIntegerv()
GL_INDEX_ARRAY_POINTER	Pointer to the index array	vertex-array	NULL	glGetPointerv()
GL_TEXTURE_COORD_ARRAY	Texture coordinate array enable	vertex-array	GL_FALSE	glIsEnabled()
GL_TEXTURE_COORD_ARRAY_SIZE	Texture coordinates per element	vertex-array	4	glGetIntegerv()

GL_TEXTURE_COORD_ARRAY_TYPE	Type of texture coordinates	vertex-array	GL_FLOAT	glGetIntegerv()
GL_TEXTURE_COORD_ARRAY_STRIDE	Stride between texture coordinates	vertex-array	0	glGetIntegerv()
GL_TEXTURE_COORD_ARRAY_POINTER	Pointer to the texture coordinate array	vertex-array	NULL	glGetPointerv()
GL_EDGE_FLAG_ARRAY	Edge flag array enable	vertex-array	GL_FALSE	glIsEnabled()
GL_EDGE_FLAG_ARRAY_STRIDE	Stride between edge flags	vertex-array	0	glGetIntegerv()
GL_EDGE_FLAG_ARRAY_POINTER	Pointer to the edge flag array	vertex-array	NULL	glGetPointerv()

Table 3 : Transformation State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_MODELVIEW_MATRIX	Modelview matrix stack	-	Identity	glGetFloatv()
GL_PROJECTION_MATRIX	Projection matrix stack	-	Identity	glGetFloatv()
GL_TEXTURE_MATRIX	Texture matrix stack	-	Identity	glGetFloatv()
GL_VIEWPORT	Viewport origin and extent	viewport	-	glGetIntegerv()
GL_DEPTH_RANGE	Depth range near and far	viewport	0, 1	glGetFloatv()
GL_MODELVIEW_STACK_DEPTH	Modelview matrix stack pointer	-	1	glGetIntegerv()
GL_PROJECTION_STACK_DEPTH	Projection matrix stack pointer	-	1	glGetIntegerv()
GL_TEXTURE_STACK_DEPTH	Texture matrix stack pointer	-	1	glGetIntegerv()
GL_MATRIX_MODE	Current matrix mode	transform	GL_MODELVIEW	glGetIntegerv()
GL_NORMALIZE	Current normal normalization	transform/	GL_FALSE	glIsEnabled()

	on/off	enable		
GL_CLIP_PLANE*i*	User clipping plane coefficients	transform	0, 0, 0, 0	glGetClipPlane ()
GL_CLIP_PLANE*i*	*i*th user clipping plane enabled	transform/ enable	GL_FALSE	glIsEnabled()

Table 4 : Coloring State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_FOG_COLOR	Fog color	fog	0, 0, 0, 0	glGetFloatv()
GL_FOG_INDEX	Fog index	fog	0	glGetFloatv()
GL_FOG_DENSITY	Exponential fog density	fog	1.0	glGetFloatv()
GL_FOG_START	Linear fog start	fog	0.0	glGetFloatv()
GL_FOG_END	Linear fog end	fog	1.0	glGetFloatv()
GL_FOG_MODE	Fog mode	fog	GL_EXP	glGetIntegerv()
GL_FOG	True if fog enabled	fog/enable	GL_FALSE	glIsEnabled()
GL_SHADE_MODEL	glShadeModel() setting	Lighting	GL_SMOOTH	glGetIntegerv()

Table 5 : (continued) Lighting State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_LIGHTING	True if lighting is enabled	lighting/e nable	GL_FALSE	glIsEnabled()
GL_COLOR_MATERIAL	True if color tracking is enabled	lighting	GL_FALSE	glIsEnabled()
GL_COLOR_MATERIAL_PARAMETE R	Material properties tracking current color	lighting	GL_AMBIENT_ AND_DIFFUSE	glGetIntegerv()
GL_COLOR_MATERIAL_FACE	Face(s) affected by color tracking	lighting	GL_FRONT_ AND_BACK	glGetIntegerv()
GL_AMBIENT	Ambient material color	lighting	(0.2, 0.2, 0.2, 1.0)	glGetMaterialfv()
GL_DIFFUSE	Diffuse material color	lighting	(0.8, 0.8, 0.8, 1.0)	glGetMaterialfv()
GL_SPECULAR	Specular	lighting	(0.0, 0.0, 0.0,	glGetMaterialfv()

	material color		1.0)	
GL_EMISSION	Emissive material color	lighting	(0.0, 0.0, 0.0, 1.0)	glGetMaterialfv()
GL_SHININESS	Specular exponent of material	lighting	0.0	glGetMaterialfv()
GL_LIGHT_MODEL_AMBIENT	Ambient scene color	lighting	(0.2, 0.2, 0.2, 1.0)	glGetFloatv()
GL_LIGHT_MODEL_LOCAL_VIEWER	Viewer is local	lighting	GL_FALSE	glGetBooleanv()
GL_LIGHT_MODEL_TWO_SIDE	Use two-sided lighting	lighting	GL_FALSE	glGetBooleanv()
GL_AMBIENT	Ambient intensity of light *i*	lighting	(0.0,0.0,0.0,1.0)	glGetLightfv()
GL_DIFFUSE	Diffuse intensity of light *i*	lighting	-	glGetLightfv()
GL_SPECULAR	Specular intensity of light *i*	lighting	-	glGetLightfv()
GL_POSITION	Position of light *i*	lighting	(0.0, 0.0, 1.0, 0.0)	glGetLightfv()
GL_CONSTANT_ATTENUATION	Constant attenuation factor	lighting	1.0	glGetLightfv()
GL_LINEAR_ATTENUATION	Linear attenuation factor	lighting	0.0	glGetLightfv()
GL_QUADRATIC_ATTENUATION	Quadratic attenuation factor	lighting	0.0	glGetLightfv()
GL_SPOT_DIRECTION	Spotlight direction of light *i*	lighting	(0.0, 0.0, -1.0)	glGetLightfv()
GL_SPOT_EXPONENT	Spotlight exponent of light *i*	lighting	0.0	glGetLightfv()
GL_SPOT_CUTOFF	Spotlight angle of light *i*	lighting	180.0	glGetLightfv()
GL_LIGHT*i*	True if light *i* enabled	lighting/enable	GL_FALSE	glIsEnabled()
GL_COLOR_INDEXES	ca, cd, and cs	lighting/e	0, 1, 1	glGetMaterialfv()

	for color-index lighting	nable		

Table 6 : (continued) Rasterization State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_POINT_SIZE	Point size	point	1.0	glGetFloatv()
GL_POINT_SMOOTH	Point antialiasing on	point/enable	GL_FALSE	glIsEnabled()
GL_LINE_WIDTH	Line width	line	1.0	glGetFloatv()
GL_LINE_SMOOTH	Line antialiasing on	line/enable	GL_FALSE	glIsEnabled()
GL_LINE_STIPPLE_PATTERN	Line stipple	line	1's	glGetIntegerv()
GL_LINE_STIPPLE_REPEAT	Line stipple repeat	line	1	glGetIntegerv()
GL_LINE_STIPPLE	Line stipple enable	line/enable	GL_FALSE	glIsEnabled()
GL_CULL_FACE	Polygon culling enabled	polygon/enable	GL_FALSE	glIsEnabled()
GL_CULL_FACE_MODE	Cull front-/back-facing polygons	polygon	GL_BACK	glGetIntegerv()
GL_FRONT_FACE	Polygon front-face CW/CCW indicator	polygon	GL_CCW	glGetIntegerv()
GL_POLYGON_SMOOTH	Polygon antialiasing on	polygon/enable	GL_FALSE	glIsEnabled()
GL_POLYGON_MODE	Polygon rasterization mode (front and back)	polygon	GL_FILL	glGetIntegerv()
GL_POLYGON_OFFSET_FACTOR	Polygon offset factor	polygon	0	glGetFloatv()
GL_POLYGON_OFFSET_BIAS	Polygon offset bias	polygon	0	glGetFloatv()
GL_POLYGON_OFFSET_POINT	Polygon offset enable for GL_POINT mode rasterization	polygon/enable	GL_FALSE	glIsEnabled()
GL_POLYGON_OFFSET_LINE	Polygon offset enable for GL_LINE mode rasterization	polygon/enable	GL_FALSE	glIsEnabled()
GL_POLYGON_OFFSET_FILL	Polygon offset enable for GL_FILL mode rasterization	polygon/enable	GL_FALSE	glIsEnabled()
GL_POLYGON_STIPPLE	Polygon stipple enable	polygon/enable	GL_FALSE	glIsEnabled()

-	Polygon stipple pattern	polygon-stipple	1's	glGetPolygon-Stipple()

Table 7 : (continued) Texturing State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_TEXTURE_*x*	True if *x*-D texturing enabled (*x* is 1D or 2D)	texture/enable	GL_FALSE	glIsEnabled()
GL_TEXTURE_BINDING_*x*	Texture object bound to GL_TEXTURE_*x* (*x* is 1D or 2D)	texture	GL_FALSE	glGetIntegerv()
GL_TEXTURE	*x*-D texture image at level of detail *i*	-	-	glGetTexImage()
GL_TEXTURE_WIDTH	*x*-D texture image *i*'s width	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_HEIGHT	*x*-D texture image *i*'s height	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_BORDER	*x*-D texture image *i*'s border width	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_INTERNAL_FORMAT	*x*-D texture image *i*'s internal image format	-	1	glGetTexLevelParameter*( )
GL_TEXTURE_RED_SIZE	*x*-D texture image *i*'s red resolution	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_GREEN_SIZE	*x*-D texture image *i*'s green resolution	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_BLUE_SIZE	*x*-D texture image *i*'s blue resolution	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_ALPHA_SIZE	*x*-D texture image *i*'s alpha resolution	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_LUMINANCE_SIZE	*x*-D texture image *i*'s	-	0	glGetTexLevelParameter*( )

	luminance resolution			
GL_TEXTURE_INTENSITY_SIZE	*x*-D texture image *i*'s intensity resolution	-	0	glGetTexLevelParameter*( )
GL_TEXTURE_BORDER_COLOR	Texture border color	texture	0, 0, 0, 0	glGetTexParameter*()
GL_TEXTURE_MIN_FILTER	Texture minification function	texture	GL_ NEAREST_ MIPMAP_ LINEAR	glGetTexParameter*()
GL_TEXTURE_MAG_FILTER	Texture magnification function	texture	GL_LINEA R	glGetTexParameter*()
GL_TEXTURE_WRAP_*x*	Texture wrap mode (*x* is S or T)	texture	GL_REPEAT	glGetTexParameter*()
GL_TEXTURE_PRIORITY	Texture object priority	texture	1	glGetTexParameter*()
GL_TEXTURE_RESIDENCY	Texture residency	texture	GL_FALSE	glGetTexParameteriv()
GL_TEXTURE_ENV_MODE	Texture application function	texture	GL_ MODULATE	glGetTexEnviv()
GL_TEXTURE_ENV_COLOR	Texture environment color	texture	0, 0, 0, 0	glGetTexEnvfv()
GL_TEXTURE_GEN_*x*	Texgen enabled (*x* is S, T, R, or Q)	texture/e nable	GL_FALSE	glIsEnabled()
GL_EYE_PLANE	Texgen plane equation coefficients	texture	-	glGetTexGenfv()
GL_OBJECT_PLANE	Texgen object linear coefficients	texture	-	glGetTexGenfv()
GL_TEXTURE_GEN_MODE	Function used for texgen	texture	GL_EYE_ LINEAR	glGetTexGeniv()

Table 8 : (continued) Pixel Operations

State Variable	Description	Attribute Group	Initial Value	Get Command

GL_SCISSOR_TEST	Scissoring enabled	scissor/enable	GL_FALSE	glIsEnabled()
GL_SCISSOR_BOX	Scissor box	scissor	-	glGetIntegerv()
GL_ALPHA_TEST	Alpha test enabled	color-buffer/ enable	GL_FALSE	glIsEnabled()
GL_ALPHA_TEST_FUNC	Alpha test function	Color-buffer	GL_ALWAYS	glGetIntegerv()
GL_ALPHA_TEST_REF	Alpha test reference value	Color-buffer	0	glGetIntegerv()
GL_STENCIL_TEST	Stenciling enabled	stencil-buffer/en able	GL_FALSE	glIsEnabled()
GL_STENCIL_FUNC	Stencil function	stencil-buffer	GL_ALWAYS	glGetIntegerv()
GL_STENCIL_VALUE_MASK	Stencil mask	stencil-buffer	1's	glGetIntegerv()
GL_STENCIL_REF	Stencil reference value	stencil-buffer	0	glGetIntegerv()
GL_STENCIL_FAIL	Stencil fail action	stencil-buffer	GL_KEEP	glGetIntegerv()
GL_STENCIL_PASS_DEPTH_FAIL	Stencil depth buffer fail action	stencil-buffer	GL_KEEP	glGetIntegerv()
GL_STENCIL_PASS_DEPTH_PASS	Stencil depth buffer pass action	stencil-buffer	GL_KEEP	glGetIntegerv()
GL_DEPTH_TEST	Depth buffer enabled	depth- buffer/ena ble	GL_FALSE	glIsEnabled()
GL_DEPTH_FUNC	Depth buffer test function	depth-buffer	GL_LESS	glGetIntegerv()
GL_BLEND	Blending enabled	color-buffer/ enable	GL_FALSE	glIsEnabled()
GL_BLEND_SRC	Blending source function	color-buffer	GL_ONE	glGetIntegerv()
GL_BLEND_DST	Blending destination function	color-buffer	GL_ZERO	glGetIntegerv()
GL_DITHER	Dithering enabled	color-buffer/ enable	GL_TRUE	glIsEnabled()
GL_INDEX_LOGIC_OP	Color index logical operation enabled	color-buffer/ enable	GL_FALSE	glIsEnabled()
GL_COLOR_LOGIC_OP	RGBA color logical operation enabled	color-buffer/ enable	GL_FALSE	glIsEnabled()
GL_LOGIC_OP_MODE	Logical operation function	color-buffer	GL_COPY	glGetIntegerv()

Table 9 : Framebuffer Control State Variables				
State Variable	Description	Attribute Group	Initial Value	Get Command
GL_DRAW_BUFFER	Buffers selected for drawing	color-buffer	-	glGetIntegerv()
GL_INDEX_WRITEMASK	Color-index writemask	color-buffer	1's	glGetIntegerv()
GL_COLOR_WRITEMASK	Color write enables; R, G, B, or A	color-buffer	GL_TRUE	glGetBooleanv()
GL_DEPTH_WRITEMASK	Depth buffer enabled for writing	depth-buffer	GL_TRUE	glGetBooleanv()
GL_STENCIL_WRITEMASK	Stencil-buffer writemask	stencil-buffer	1's	glGetIntegerv()
GL_COLOR_CLEAR_VALUE	Color-buffer clear value (RGBA mode)	color-buffer	0, 0, 0, 0	glGetFloatv()
GL_INDEX_CLEAR_VALUE	Color-buffer clear value (color-index mode)	color-buffer	0	glGetFloatv()
GL_DEPTH_CLEAR_VALUE	Depth-buffer clear value	depth-buffer	1	glGetIntegerv()
GL_STENCIL_CLEAR_VALUE	Stencil-buffer clear value	stencil-buffer	0	glGetIntegerv()
GL_ACCUM_CLEAR_VALUE	Accumulation-buffer clear value	accum-buffer	0	glGetFloatv()

Table 10 : (continued) Pixel State Variables				
State Variable	Description	Attribute Group	Initial Value	Get Command
GL_UNPACK_SWAP_BYTES	Value of GL_UNPACK_SWAP_BYTES	pixel-store	GL_FALSE	glGetBooleanv()
GL_UNPACK_LSB_FIRST	Value of GL_UNPACK_LSB_FIRST	pixel-store	GL_FALSE	glGetBooleanv()
GL_UNPACK_ROW_LENGTH	Value of GL_UNPACK_ROW_LENGTH	pixel-store	0	glGetIntegerv()
GL_UNPACK_SKIP_ROWS	Value of GL_UNPACK_SKIP_ROWS	pixel-store	0	glGetIntegerv()
GL_UNPACK_SKIP_PIXELS	Value of GL_UNPACK_SKIP_PIXELS	pixel-store	0	glGetIntegerv()
GL_UNPACK_ALIGNMENT	Value of GL_UNPACK_ALIGNMENT	pixel-store	4	glGetIntegerv()
GL_PACK_SWAP_BYTES	Value of GL_PACK_SWAP_BYTES	pixel-store	GL_FALSE	glGetBooleanv()
GL_PACK_LSB_FIRST	Value of	pixel-	GL_FALSE	glGetBooleanv()

	GL_PACK_LSB_FIRST	store		
GL_PACK_ROW_LENGTH	Value of GL_PACK_ROW_LENGTH	pixel-store	0	glGetIntegerv()
GL_PACK_SKIP_ROWS	Value of GL_PACK_SKIP_ROWS	pixel-store	0	glGetIntegerv()
GL_PACK_SKIP_PIXELS	Value of GL_PACK_SKIP_PIXELS	pixel-store	0	glGetIntegerv()
GL_PACK_ALIGNMENT	Value of GL_PACK_ALIGNMENT	pixel-store	4	glGetIntegerv()
GL_MAP_COLOR	True if colors are mapped	pixel	GL_FALSE	glGetBooleanv()
GL_MAP_STENCIL	True if stencil values are mapped	pixel	GL_FALSE	glGetBooleanv()
GL_INDEX_SHIFT	Value of GL_INDEX_SHIFT	pixel	0	glGetIntegerv()
GL_INDEX_OFFSET	Value of GL_INDEX_OFFSET	pixel	0	glGetIntegerv()
GL_*x*_SCALE	Value of GL_*x*_SCALE; *x* is GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, or GL_DEPTH	pixel	1	glGetFloatv()
GL_*x*_BIAS	Value of GL_*x*_BIAS; *x* is one of GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, or GL_DEPTH	pixel	0	glGetFloatv()
GL_ZOOM_X	*x* zoom factor	pixel	1.0	glGetFloatv()
GL_ZOOM_Y	*y* zoom factor	pixel	1.0	glGetFloatv()
GL_x	glPixelMap() translation tables; *x* is a map name	-	0's	glGetPixelMap*()
GL_*x*_SIZE	Size of table *x*	-	1	glGetIntegerv()
GL_READ_BUFFER	Read source buffer	pixel	-	glGetIntegerv()

Table 11 : Evaluator State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_ORDER	1D map order	-	1	glGetMapiv()
GL_ORDER	2D map orders	-	1, 1	glGetMapiv()
GL_COEFF	1D control points	-	-	glGetMapfv()

GL_COEFF	2D control points	-	-	glGetMapfv()
GL_DOMAIN	1D domain endpoints	-	-	glGetMapfv()
GL_DOMAIN	2D domain endpoints	-	-	glGetMapfv()
GL_MAP1_*x*	1D map enables: *x* is map type	eval/enable	GL_FALSE	glIsEnabled()
GL_MAP2_*x*	2D map enables: *x* is map type	eval/enable	GL_FALSE	glIsEnabled()
GL_MAP1_GRID_DOMAIN	1D grid endpoints	eval	0, 1	glGetFloatv()
GL_MAP2_GRID_DOMAIN	2D grid endpoints	eval	0, 1; 0, 1	glGetFloatv()
GL_MAP1_GRID_SEGMENTS	1D grid divisions	eval	1	glGetFloatv()
GL_MAP2_GRID_SEGMENTS	2D grid divisions	eval	1,1	glGetFloatv()
GL_AUTO_NORMAL	True if automatic normal generation enabled	eval	GL_FALSE	glIsEnabled()

Table 12 : Hint State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_PERSPECTIVE_CORRECTION_HINT	Perspective correction hint	hint	GL_DONT_CARE	glGetIntegerv()
GL_POINT_SMOOTH_HINT	Point smooth hint	hint	GL_DONT_CARE	glGetIntegerv()
GL_LINE_SMOOTH_HINT	Line smooth hint	hint	GL_DONT_CARE	glGetIntegerv()
GL_POLYGON_SMOOTH_HINT	Polygon smooth hint	hint	GL_DONT_CARE	glGetIntegerv()
GL_FOG_HINT	Fog hint	hint	GL_DONT_CARE	glGetIntegerv()

Table 13 : (continued) Implementation-Dependent State Variables

State Variable	Description	Attribute Group	Minimum Value	Get Command
GL_MAX_LIGHTS	Maximum number of lights	-	8	glGetIntegerv()
GL_MAX_CLIP_PLANES	Maximum number of user clipping planes	-	6	glGetIntegerv()

GL_MAX_MODELVIEW_STACK_DEPTH	Maximum modelview-matrix stack depth	-	32	glGetIntegerv()
GL_MAX_PROJECTION_STACK_DEPTH	Maximum projection-matrix stack depth	-	2	glGetIntegerv()
GL_MAX_TEXTURE_STACK_DEPTH	Maximum depth of texture matrix stack	-	2	glGetIntegerv()
GL_SUBPIXEL_BITS	Number of bits of subpixel precision in x and y	-	4	glGetIntegerv()
GL_MAX_TEXTURE_SIZE	See discussion "Texture Proxy"	-	64	glGetIntegerv()
GL_MAX_PIXEL_MAP_TABLE	Maximum size of a glPixelMap() translation table	-	32	glGetIntegerv()
GL_MAX_NAME_STACK_DEPTH	Maximum selection-name stack depth	-	64	glGetIntegerv()
GL_MAX_LIST_NESTING	Maximum display-list call nesting	-	64	glGetIntegerv()
GL_MAX_EVAL_ORDER	Maximum evaluator polynomial order	-	8	glGetIntegerv()
GL_MAX_VIEWPORT_DIMS	Maximum viewport dimensions	-	-	glGetIntegerv()
GL_MAX_ATTRIB_STACK_DEPTH	Maximum depth of the attribute stack	-	16	glGetIntegerv()
GL_MAX_CLIENT_ATTRIB_STACK_DEPTH	Maximum depth of the client attribute stack	-	16	glGetIntegerv()
GL_AUX_BUFFERS	Number of auxiliary buffers	-	0	glGetBooleanv()
GL_RGBA_MODE	True if color buffers store	-	-	glGetBooleanv()

	RGBA			
GL_INDEX_MODE	True if color buffers store indices	-	-	glGetBooleanv()
GL_DOUBLEBUFFER	True if front and back buffers exist	-	-	glGetBooleanv()
GL_STEREO	True if left and right buffers exist	-	-	glGetBooleanv()
GL_POINT_SIZE_RANGE	Range (low to high) of antialiased point sizes	-	1, 1	glGetFloatv()
GL_POINT_SIZE_GRANULARITY	Antialiased point-size granularity	-	-	glGetFloatv()
GL_LINE_WIDTH_RANGE	Range (low to high) of antialiased line widths	-	1, 1	glGetFloatv()
GL_LINE_WIDTH_GRANULARITY	Antialiased line-width granularity	-	-	glGetFloatv()

Table 14 : Implementation-Dependent Pixel-Depth State Variables (continued)

State Variable	Description	Attribute Group	Minimum Value	Get Command
GL_RED_BITS	Number of bits per red component in color buffers	-	-	glGetIntegerv()
GL_GREEN_BITS	Number of bits per green component in color buffers	-	-	glGetIntegerv()
GL_BLUE_BITS	Number of bits per blue component in color buffers	-	-	glGetIntegerv()
GL_ALPHA_BITS	Number of bits per alpha component in color buffers	-	-	glGetIntegerv()
GL_INDEX_BITS	Number of bits per index in color buffers	-	-	glGetIntegerv()
GL_DEPTH_BITS	Number of depth-buffer bitplanes	-	-	glGetIntegerv()
GL_STENCIL_BITS	Number of stencil bitplanes	-	-	glGetIntegerv()

GL_ACCUM_RED_BITS	Number of bits per red component in the accumulation buffer	-	-	glGetIntegerv()
GL_ACCUM_GREEN_BITS	Number of bits per green component in the accumulation buffer	-	-	glGetIntegerv()
GL_ACCUM_BLUE_BITS	Number of bits per blue component in the accumulation buffer	-	-	glGetIntegerv()
GL_ACCUM_ALPHA_BITS	Number of bits per alpha component in the accumulation buffer	-	-	glGetIntegerv()

Table 15 : Miscellaneous State Variables

State Variable	Description	Attribute Group	Initial Value	Get Command
GL_LIST_BASE	Setting of glListBase()	list	0	glGetIntegerv()
GL_LIST_INDEX	Number of display list under construction; 0 if none	-	0	glGetIntegerv()
GL_LIST_MODE	Mode of display list under construction; undefined if none	-	0	glGetIntegerv()
GL_ATTRIB_STACK_DEPTH	Attribute stack pointer	-	0	glGetIntegerv()
GL_CLIENT_ATTRIB_STACK_DEPTH	Client attribute stack pointer	-	0	glGetIntegerv()
GL_NAME_STACK_DEPTH	Name stack depth	-	0	glGetIntegerv()
GL_RENDER_MODE	glRenderMode() setting	-	GL_RENDER	glGetIntegerv()
GL_SELECTION_BUFFER_POINTER	Pointer to selection buffer	select	0	glGetPointerv()
GL_SELECTION_BUFFER_SIZE	Size of selection buffer	select	0	glGetIntegerv()
GL_FEEDBACK_BUFFER_POINTER	Pointer to feedback buffer	feedback	0	glGetPointerv()
GL_FEEDBACK_BUFFER_SIZE	Size of feedback buffer	feedback	0	glGetIntegerv()
GL_FEEDBACK_BUFFER_TYPE	Type of feedback	feedback	GL_2D	glGetIntegerv()

	buffer			
-	Current error code(s)	-	0	glGetError()

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glInitNames; stdcall; external 'OPENGL32.DLL';	void glInitNames(void);

توضیح :

سبب می شود تا پشته نام ها با حالت پیش فرض خود آغـاز شـود . پشـته نـام هـا در حالـت انتخاب برای اینکه مجموعه ای از دستورات ترسیمی نامی یکتا داشته باشند بکار می رود . پشـته نام ها از مجموعه ای از اعداد صحیح مرتب شده تشکیل شده است .

اگر حالت رندر کردن GL_SELECT نباشد ، پشته نام ها همواره خالی خواهد بود .

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glLoadName(name: GLuint); stdcall; external 'OPENGL32.DLL';	void glLoadName(GLuint name);

توضیح :

نامی را به درون پشته نامها بارگذاری می کند .

Name : نامی است که جایگزین بالاترین مقدار پشته نام ها خواهد شد .

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glPushName(name: GLuint); stdcall; external 'OPENGL32.DLL'; Procedure glPopName; stdcall; external 'OPENGL32.DLL';	void glPushName(GLuint name); void glPopName(void);

توضیح :

نامی را بدرون پشته نام ها قرار می دهند و برعکس .

Name : نامی است که بدرون پشته نام ها قرار داده خواهد شد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
GLint glRenderMode(GLenum mode);	Function glRenderMode(mode: GLenum): GLint; stdcall; external 'OPENGL32.DLL';

توضیح :

حالت رندر کردن را تنظیم می کند .

Mode : حالت رندر کردن می باشد . سه ثابت سمبولیک زیر در این حالت قابل قبول هستند .

GL_RENDER : حالت پیش فرض و نرمال ترسیمات در OpenGL است .

GL_SELECT : حالت انتخاب . در این حالت تعداد رکوردهای انتخاب شده به بافر انتخاب منتقل می شود .

GL_FEEDBACK : حالت feedback . در این حالت تعداد مقادیر به بافر مربوطه منتقل می شوند .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glSelectBuffer(GLsizei size,GLuint *buffer);	Procedure glSelectBuffer(size: GLsizei; buffer: PGLuint); stdcall; external 'OPENGL32.DLL';

توضیح :

بافری را (آرایه ای را) برای مقادیر حالت انتخاب برپا می سازد .

Size : اندازه بافر می باشد .

Buffer : داده های انتخاب را باز می گرداند .

این تابع باید قبل از فعال سازی حالت GL_SELECT فراخوانی شود .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void gluPickMatrix(GLdouble x,GLdouble y, GLdouble width,GLdouble height, GLint viewport[4]);	type TViewPortArray = Array[0..3] of GLint; Procedure gluPickMatrix(x: GLdouble; y: GLdouble; width: GLdouble; height: GLdouble; viewport: TViewPortArray); stdcall; external 'GLU32.DLL';

توضیح :

این تابع ناحیه انتخاب را معین می کند .

x,y : مرکز ناحیه انتخاب در مختصات پنجره .

width, height : طول و عرض ناحیه انتخاب در مختصات پنجره .

viewport : درگاه دید جاری که از تابع glGetIntegerv بدست می آید .

این تابع ترسیم را به ناحیه کوچکی از درگاه دید محدود می کند . برای اینکه مشخص کنیم کدامیک از اشیاء نزدیک کرسر ماوس ترسیم شده اند :

۱- از gluPickMatrix استفاده کنید تا ترسیم را به ناحیه کوچکی حول کرسر ماوس محدود کنید .

۲- به حالت انتخاب توسط glRenderMode وارد شوید و سپس صحنه را رندر کنید .

تمام اشیایی که نزدیک کرسر ماوس ترسیم شده اند ، مشخص شده و در بافر انتخاب ذخیره می شوند .

برنامه فصل :

برای اجرای این برنامه به یک کنترل تایمر با Interval مساوی ۶۰ نیاز دارید.

```
unit Ch17;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics,
  Controls, Forms,Dialogs , SPF , OpenGL , ExtCtrls ;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;

  f_Hdc : LongInt;

implementation
{$R *.dfm}
const
 EARTH =101 ;// This is the object ID for the EARTH
```

```
 SUN  =100 ; // This is the object ID for the SUN
 PLUTO =102 ;// This is the object ID for PLUTO
var
 SunRotation : single   = 90;// This holds our sun's current rotation
 EarthRotation :single = 90; // This holds our earth's current rotation
 PlutoRotation : single = 90;// This holds pluto's current rotation
 pObj  : GLUquadricObj; // Storage For Our Quadratic Objects
    ambient : array[0..3] of GLfloat=( 0, 0, 0, 1 );
    diffuse : array[0..3] of GLfloat=( 1, 1, 1, 1 );
    specular : array[0..3] of GLfloat=( 1, 1, 1, 1 );
    position :array[0..3] of GLfloat=( 0, 3, 2, 0 );
    lmodel_ambient :array[0..3] of GLfloat=( 0.4, 0.4, 0.4, 1 );
    local_view : array[0..0] of GLfloat=(0);

    no_mat :array[0..3] of GLfloat=( 0, 0, 0, 1 );
    mat_diffuse  :array[0..3] of GLfloat=(0.1, 0.5, 0.4, 1);
    no_shininess : array[0..0] of GLfloat=(0);

    mat_ambient_color1 :array[0..3] of GLfloat=(0.2, 0.8, 0.2, 1);
    mat_emission1 :array[0..3] of GLfloat=( 0.3, 0.2, 0.2, 0 );

    mat_ambient_color :array[0..3] of GLfloat=(0.1, 0.4, 0.1, 1);
    mat_emission :array[0..3] of GLfloat=( 0.5, 0.1, 0.1, 0 );

//////////////////////////// INITIALIZE GL \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*
///
///       This function handles all the initialization for OpenGL.
///
//////////////////////////// INITIALIZE GL \\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\*

procedure InitGL();
begin
  glClearColor (0, 0.1, 0.1, 0);
  glEnable(GL_DEPTH_TEST);     // Enable Depth Testing

   glLightfv (GL_LIGHT0, GL_AMBIENT, @ambient);
   glLightfv (GL_LIGHT0, GL_DIFFUSE, @diffuse);
   glLightfv (GL_LIGHT0, GL_POSITION, @position);
   glLightModelfv (GL_LIGHT_MODEL_Ambient, @lmodel_ambient);
   glLightModelfv (GL_LIGHT_MODEL_LOCAL_VIEWER, @local_view);

   glEnable (GL_LIGHTING);
   glEnable (GL_LIGHT0);
end;

procedure RenderScene();
begin
    // Clear The Screen And The Depth Buffer
    glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();        // Reset The matrix
    gluLookAt(0, 3, 6,     0, 0, 0,     0, 1, 0);
    // This clears the name stack so we always start with 0 names.
    glInitNames();
    // This next line is important.  If you don't push on at least ONE name,
    // The selection won't work.  Instead of glLoadName()/glEnd() you can use
    // glPushName(TheID) and glPopName();  Then you don't need to glPushName(0);
    glPushName(0);    // This starts off the first object in the stack
    // Get a new Quadric off the stack
```

```
 pObj := gluNewQuadric();
 glLoadName(SUN);    // Push on our SUN label (IMPORTANT)
 glBegin(GL_LINES);
 glEnd();
 // Here we push on a new matrix so we don't affect any other quadrics.
 glPushMatrix();                  // Push on a new matrix scope
       glTranslatef(0, 0, 0);     // Translate this sphere to the left
    // Rotate the sphere around the Y axis to make it spin
       glRotatef(SunRotation, 0, 1.0, 0);

    glMaterialfv (GL_FRONT, GL_AMBIENT, @no_mat);
    glMaterialfv (GL_FRONT, GL_DIFFUSE, @mat_diffuse);
    glMaterialfv (GL_FRONT, GL_SPECULAR, @no_mat);
    glMaterialfv (GL_FRONT, GL_SHININESS, @no_shininess);
    glMaterialfv (GL_FRONT, GL_EMISSION, @no_mat);

       gluSphere(pObj, 0.5, 20, 20); // Draw the sunwith a radius of 0.5
 glPopMatrix();          // End the current scope of this matrix
glEnd(); // Stop assigning polygons to the SUN label (IMPORTANT)

glLoadName(EARTH); // Push on our EARTH label (IMPORTANT)

glBegin(GL_LINES);
glEnd();

glPushMatrix();          // Push on a new matrix scope
  // Rotate the sphere around the origin (the sun)
  glRotatef(EarthRotation / 3, 0, 1.0, 0);
  glTranslatef(-2, 0, 0);          // Translate this sphere to the left
  glRotatef(EarthRotation, 0, 1.0, 0); // Rotate the sphere to make it spin

  glMaterialfv (GL_FRONT, GL_AMBIENT, @mat_ambient_color);
  glMaterialfv (GL_FRONT, GL_DIFFUSE, @mat_diffuse);
  glMaterialfv (GL_FRONT, GL_SPECULAR, @no_mat);
  glMaterialfv (GL_FRONT, GL_SHININESS, @no_shininess);
  glMaterialfv (GL_FRONT, GL_EMISSION, @mat_emission);

  // Draw the sphere with a radius of 0.2 so it's smaller than the sun
  gluSphere(pObj, 0.2, 20, 20);
glPopMatrix(); // End the current scope of this matrix
 glEnd();       // Stop assigning polygons to the EARTH label (IMPORTANT)

 glLoadName(PLUTO);          // Push on our PLUTO label (IMPORTANT)

 glBegin(GL_LINES);
 glEnd();

 glPushMatrix(); // Push on a new matrix scope
    // Rotate the sphere around the sun
    glRotatef(PlutoRotation / 2, 0, 1.0, 0);
    // Translate this sphere farther away from the sun than the earth
    glTranslatef(3, 0, 0);
    // Rotate the sphere around itself to produce the spin
    glRotatef(PlutoRotation, 0, 1.0, 0);

  glMaterialfv (GL_FRONT, GL_AMBIENT, @mat_ambient_color1);
  glMaterialfv (GL_FRONT, GL_DIFFUSE, @mat_diffuse);
  glMaterialfv (GL_FRONT, GL_SPECULAR, @no_mat);
```

```
    glMaterialfv (GL_FRONT, GL_SHININESS, @no_shininess);
    glMaterialfv (GL_FRONT, GL_EMISSION, @mat_emission1);

      // Draw the sphere with a radius of 0.1 (smallest planet)
      gluSphere(pObj, 0.1, 20, 20);
      glPopMatrix();        // End the current scope of this matrix
    glEnd(); // Stop assigning polygons to our PLUTO label (IMPORTANT)

    SwapBuffers(f_Hdc);   // Swap the backbuffers to the foreground
    gluDeleteQuadric(pObj);       // Free the Quadric
   // Below we increase the rotations for each sphere.
   SunRotation   :=SunRotation+ 2.2;// Rotate the sun slowly
   EarthRotation :=SunRotation+ 2.5;// Increase the rotation for the each
   PlutoRotation :=SunRotation+ 2.6;// Make pluto go the fastest
end;

//////////////////////////////// PROCESS SELECTION \\\\\\\\\\\
//
// This takes the cursor's x and y position
// and returns the closet object
//
//////////////////////////////// PROCESS SELECTION \\\\\\\\\\\
function RetrieveObjectID( x, y : LongInt;
                      height,width: integer ): LongInt;
var
  // This will hold the amount of objects clicked
  objectsFound , i : LongInt;
  // We need an array to hold our view port coordinates
  viewportCoords : array[0..3] of  Gluint;
 // This will hold the ID's of the objects we click on.
 // We make it an arbitrary number of 32
  selectBuffer : array[0..31] of Gluint;
  lowestDepth , selectedObject : Gluint;
begin
   // glSelectBuffer is what we register our selection buffer with.
   // Setup our selection buffer to accept object ID's

   glSelectBuffer(32, @selectBuffer);
   // This function returns information about many things in OpenGL.
   //  We pass in GL_VIEWPORT
   // to get the view port coordinates.
   glGetIntegerv(GL_VIEWPORT, @viewportCoords);

   // Now we want to get out of our GL_MODELVIEW matrix and
   // start effecting our GL_PROJECTION matrix.
   // This allows us to check our X and Y coords against 3D space.
   glMatrixMode(GL_PROJECTION);
   // We push on a new matrix so we don't effect our 3D projection
   glPushMatrix();
       // Allows us to render the objects, but not change the frame buffer
          glRenderMode(GL_SELECT);
          glLoadIdentity(); // Reset our projection matrix
          // gluPickMatrix allows us to create a projection
       // matrix that is around our cursor.
       // NOTE : y has -27 to account to caption bar
       gluPickMatrix(x, form1.Height - y-27, 2, 2, @viewportCoords); //or
//          gluPickMatrix(x, viewportCoords[3] - y, 2, 2, @viewportCoords);
          // Next, we just call our normal gluPerspective() function,
```

```
      //exactly as we did on startup.
         // This is to multiply the perspective matrix by the
      //pick matrix we created up above.

      gluPerspective(45.0, viewportCoords[2]/viewportCoords[3], 1.0, 100.0);

         glMatrixMode(GL_MODELVIEW); // Go back into our model view matrix
      // Now we render into our selective mode to pinpoint clicked objects
         RenderScene();
      // If we return to our normal render mode from select mode,
      //glRenderMode returns
         // the number of objects that were found in our specified
      // region (specified in gluPickMatrix())
      objectsFound := glRenderMode(GL_RENDER);

      // Put our projection matrix back to normal.
         glMatrixMode(GL_PROJECTION);
   glPopMatrix();          // Stop effecting our projection matrix
    glMatrixMode(GL_MODELVIEW);
   //objectsFound should be at least 1 if we found an object.

         if (objectsFound > 0)     then
      begin
         // If we found more than one object, we need to check the depth values
         // of all the objects found.  The object with the LEAST depth value is
         // the closest object that we clicked on.
         lowestDepth := selectBuffer[1];
         selectedObject := selectBuffer[3];
         // Go through all of the objects found, but start at the second one
          for  i := 1  to objectsFound-1 do
          begin
          // Check if the current objects depth is lower
          //than the current lowest
          // Notice we times i by 4 (4 values for each object)
          // and add 1 for the depth.
                  if(selectBuffer[(i * 4) + 1] < lowestDepth) then
                    begin
                       // Set the current lowest depth
                       lowestDepth := selectBuffer[(i * 4) + 1];
                       // Set the current object ID
                       selectedObject := selectBuffer[(i * 4) + 3];
                    end;
             end;
                  // Return the selected object
                  RetrieveObjectID := selectedObject;
            exit;
         end;
         // We didn't click on any objects so return 0
         RetrieveObjectID := 0;
end;

procedure TForm1.FormResize(Sender: TObject);
begin
   wglMakeCurrent(f_Hdc,hrc); //activate the RC
   if (height=0) then   // Prevent A Divide By Zero error
         height:=1;     // Make the Height Equal One
   glViewport(0,0,width,height); // Make our viewport the whole window
   glMatrixMode(GL_PROJECTION);// Select The Projection Matrix
```

```
   glLoadIdentity();           // Reset The Projection Matrix
   // Calculate The Aspect Ratio Of The Window
   gluPerspective(45.0,width/height,0.1,450.0);
   glMatrixMode(GL_MODELVIEW);  // Select The Modelview Matrix
   glLoadIdentity();            // Reset The Modelview Matrix
   InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
 CleanUp(f_Hdc);// Clean up and terminate.
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 f_Hdc := GetDC(handle);
 SetDCPixelFormat(f_Hdc,16,16);// Create a rendering context.
 InitGL();
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
 wglMakeCurrent(f_Hdc,hrc); //activate the RC
 RenderScene;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Application.ProcessMessages; //DoEvents
  InvalidateRect(Handle, nil, False);//   DrawScene();
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
  var
    objectID : Integer;
begin
  // Here we pass in the cursors X and Y coordinates to
  //test for an object under the mouse.
    objectID := RetrieveObjectID(x,y,height,width);
  // Now we just do a switch on our object ID's to see if we hit one.
  // Check the objectID passed back
    form1.Resize;
  if objectID= SUN then
                    MessageBox(handle, 'The Sun!', 'Click', MB_OK);
  if objectID= EARTH then         // We hit the Earth!
                    MessageBox(handle, 'The Earth!', 'Click', MB_OK);
  if objectID= PLUTO then         // We hit Pluto!
                    MessageBox(handle, 'Pluto!', 'Click', MB_OK);


end;

end.
```