**PLEASE NOTE LIBRARY PATHS ARE RELATIVE TO MY MACHINE.**

**Abstract**

For my project I decided to do a game. The game’s main protagonist is a marble which rolls around the scene. My game implements a method for loading mesh and texture data, as well as an engine to display the rendered data. Collision, gravity and acceleration have also been implemented, as well as a black lightnin.

**I. Source.cpp**

This is the file that contains the main function which sets up glut and runs the game. This file also controls certain aspects of the game that the Level class (discussed in detail later) does not.

It is of note that the Level and player are declared here, this is to facilitate between the changing of levels. A Level class is initialized and a subclass (Level1 class) is initialized. The project unfortunately only contains one level, but switching between levels is quite possible as the Level class is never initialized as an object by itself, it always points to one of its subclasses. The “Level” class is the basic engine and is not a game level in itself, although it would be possible to implement a very barebones level by using this class. This is not recommended, as then level switching would not be possible. There are two special keyboard functions of note. The “special” function sets the left, right, up, and down boolean values to true. The “specialUp” keyboard function sets the left, right, up, down Boolean values to false. This way the (player inputted) movement of the marble is controlled by a “glutTimerFunc” called “moveTimer.” “moveTimer” moves the marble if the “canMove” property is enabled. It does this by checking which of up, left, right, down are true and adjusts the rotation of the marble accordingly; It is worth noting that backface culling is enabled.

**II. Level class**

The level class does most of the work when it comes to displaying the in-game content. Firstly, the level class can load .obj files, which is very useful as models created in blender can be loaded.

Point Structure

The point structure contains the x, y, and z coordinates of the point, as well as the texture coordinates of that particular point.

Triangle Structure

The triangle structure contains 4 points and an int. The int I used to discern which normal belongs to it; it is temporality used when loading an object. The 4 points are the x, y, and z coordinates as well as the normal. The normal is a vector, and not a point, but as we know, a vector is represented is the same way as a point is

.obj Loading

A loader for .obj files was implemented from scratch. This is contained in the Level class and is defined by the function “loadObj”. loadObj takes a c++ vector class of type “Triangle” (Defined in the Level class, it essentially represents a triangle which also stores its own normal) and the address of the .obj file to be loaded. The function first loads the file, it then parses through it. It reads the vertexes into a temporary vector class of type “Point” (defined in the Level class, it represents the x, y and z co-ordinates of a point) points. Then the texture co-ordinates are loaded into a temporary vector class of type “Point.” The normals are loaded into a temporary vector class of type “Point.” Then the faces are loaded. .obj files list faces as “[vertex 1 id]/[texture coordinate 1 id]/[face id] [vertex 2 id]/[texture coordinate 2 id]/[face id] [vertex 3 id]/[texture coordinate 3 id]/[face id]” (eg. 1/2/1 2/3/1 4/4/1), these are references to the previously recorded vertices and texture coordinates, and the face number . This is why the previous data was loaded into its own vector classes. When the faces are loaded, the .obj loaded looks up which vertex is being referenced and which texture coordinate is being referenced. The face number is then recorded so that the normal can later be loaded. The normal are inputted into the triangles after the other data is loaded as the faces are triangulated and thus faces are composed of multiple triangles. Since each normal corresponds to a face in numbered order (i.e. first normal that appears is the normal of face 1), the normals vector will have copies of the same normal so that the proper face can reference the proper normal. The completed face data is then outputted into the vector class that was inputted as a parameter. It is also worth noting that the first time the loadObj class is run, it does not record texture coordinates. This is because the first .obj file loaded must be the mesh that specifies the level bounds (i.e. for collision purposes). Furthermore, the faces in the .obj file must be triangulated for this to work. This design path was taken because drawing quads is less efficient than drawing triangles and efficiency is everything when it comes to a 3d game. In addition, anything can be rendered out of only triangles and in this way triangulation is taken care of by the program (Blender) that created the .obj file

Texture Loading

Textures are loaded using SOIL (Simple OpenGL Image Library), it would have been possible to implement a bmp loader, but most images on the web are .jpg files and bitmaps are take up more space.

Player-Environment Collision

Collision is handled by the “collision” function by going through all of the faces and checking to see if the player is colliding with the face. This is done by getting the barycentric coordinates of the player relative to the face to see if the player is “within” the face. This actually only checks if the player is within the triangle, but not how close to the triangle the player is. So, a player at (0,2000,0) is within triangle (-1,0,-1), (1,0,0),(-1,0,1). The distance is then calculated by using the mathematic formula for obtaining the distance between a point and a plane. The player is only a point, and the sphere is rendered around it, so the player’s x and y coordinates are used. The plane is defined by the normal of the face and any point on the plane, in this case is it the first point that makes up the triangle. Thusly collision is checked. Next the normal in x and z directions are applied to the player (it is worth noting that in this implementation, x is forward and back, z is left and right, and y is up and down). Next the directional vector that was inputted is applied to the player. Thusly, when a player comes in contact with an angles object, the player object does not simply stop, it slides along the surface.

Gravity is handled by the “gravity” function. The gravity function works similarly to the collision function in that it goes through the faces and checks the distance from the player and the barycentric coordinates relative to the player. It differs in that when it detects a collision, it applies the full normal of that face, instead of just the x and z part of the normal. If no collisions happen then the player’s negative last y normal value is applied to the player (i.e. the player falls). The players last normal is used so that force is properly reflected, i.e. so that gravity is not greater that the up force exerted by a face. If a normal vector of for example -1 was applied, then the player would constantly sink into the floor and pop out of it, or possibly just and the player would sink through angled faces.

Drawing Objects

Objects are drawn with the “drawObject” class. The draw object class takes a face from the inputted object (which is represented as a c++ vector class of triangle faces). This face contains the vertices and the texture coordinates of the vertices. And so the material properties are set and the face is drawn textured. It is worth noting that drawn objects do not cause collision. Collision is caused by the bounding mesh. I would have been horribly inefficient to check for collision with every face of drawn objects. This is because to the fact that visual objects tend to be detailed and contain many, many faces. This is why in very many video games, bounding boxes are used. In this particular game, the bounding boxes are set by the collision mesh (which is the first mesh loaded).

**III. The Player Class**

The player class is fairly small. In essence, the player class is just a representation of where the player is and its rotation. The player class contains a rotation matrix and the xAngle (which is used for rotation in both the x and z axes, despite being named xAngle). The rotation matrix of the player is modified by the “moveTimer” function previously described. It is worth noting that the so-called “rotation matrix” is not a transformation matrix. It simply stores a 1 in position 0 if there will be a rotation about the x axis, and a 1 in position 2 if there will be a rotation about z axis. The 1 position is unused; however, it was left as a matrix of size 3 in case a rotation about the y axis was ever needed. The player is drawn as a sphere, and its texture is loaded by the Level class. The player’s move function moves the player in the direction of the inputted normal vector with a speed of “s + a” (speed + acceleration). The player accelerates as it moves. The player’s gravity function moves the player in the direction of the inputted normal vector with a speed of “g” (gravity).

**IV. Sound**

Sound is handled by the bass library. It is used for the background music.

**IV. Level1 Class**

The “Level1” class is a subclass of the level class, and extends it to become a full-fledged level (although being small, unfortunately). Firstly, the Level1 class initiates a phong lighting model. The Level1 class then specifies which resources are loaded. This is done by calling the “loadTex” and “LoadObj” functions defined by level. There are passed to the Level part of the class and that part handles the collision and the drawing of objects. The Level1 class focuses more on effect and events. Events are implemented as listeners, and these listeners are checked each time the level is drawn. The first listener checks to see if the default track “sundering1” should be changed to “sundering2.” The next 3 handle the drawbridge. The first stops the player and begins the drawbridge extension. The second begins to extend the drawbridge. The third listens for when the drawbridge is fully extended, and then returns control to the player. There is also a lightning effect.

**VI The Lightning class**

The lightning class is implemented to simulate a lightning strike. Random lines were not a good implementation as they made is seem as if the lightning was spinning rather than striking. The lightning class first creates a bolt of lightning which is comprised of 6 c++ vector classes of type “Point”. The first vector two classes describe the points in the lightning bolt until the first branch. The third branches off and hits the sword, the fourth keeps going. The fifth and sixth branch off of the fourth and strike the ground. The lightning then draws these vectors in the sequence 1, 2, 3 and 4, 5 and 6 by connecting the randomly generated dots at a speed of factor”. This gives the effect of a lightning strike, instead of random lines. The lightning then redefines the shape of the bolt. This give the effect of animated, striking lightning.

**Resources**

Barycentric algorithm adapted from Christer Ericson's [Real-Time Collision Detectio](http://realtimecollisiondetection.net/)n

.obj Files

* sword.Obj - <http://tf3dm.com/3d-model/spartan-sword-low-poly-10858.html>
* lowpolytree.obj - <http://tf3dm.com/3d-model/low-poly-tree-73217.html>
* pilar.obj - <http://tf3dm.com/3d-model/pillar-pilar-45905.html>

Images used for textures

* barkTex.jpg - <https://www.google.ca/search?sa=G&hl=en-CA&q=wood+pattern&tbm=isch&tbs=simg:CAQSjAEaiQELEKjU2AQaBAgFCAsMCxCwjKcIGmAKXggDEibaB9QJ2QcF2weOCQ6JBtUJkAneKe015SfXKOkn6yfqJ9oh7Df_1Jxowos9KNP_1FzPFSwNdxIVJjyB5TlmWNqNWCotm5v5txAEQAbVcfJhxt5SHCArPholZRIAMMCxCOrv4IGgoKCAgBEgReTOnODA&ved=0ahUKEwiqmJqH6rjKAhVhn4MKHdaKCEMQwg4IGSgA&biw=1920&bih=935#imgrc=khRoNZHF_ZKXdM%3A>
* Blade tex.jpg - <https://www.google.ca/search?sa=G&hl=en-CA&tbm=isch&tbs=simg:CAQSjAEaiQELEKjU2AQaAggDDAsQsIynCBpiCmAIAxIolQaZAZYBmhLsBY4G2QztBZkSsAGsNtYopjblJ94o6yfgIe016zfqNxowJ_1Q2UeCue3xaKAzllHL6X0vXqTPzG7hsit_1JkUsPIVXaCjfS0NWDj-3It3fzCeXgIAMMCxCOrv4IGgoKCAgBEgSC4nQjDA&ved=0ahUKEwiy4uar6rjKAhXKyIMKHdMAB8MQwg4IGSgA&biw=1920&bih=935#imgrc=WAj-6eOOvMv4pM%3A>
* grassTex.jpg - https://www.google.ca/search?sa=G&hl=en-CA&q=cartoon+grass+texture+seamless&tbm=isch&tbs=simg:CAQSiQEahgELEKjU2AQaAggEDAsQsIynCBpfCl0IAxIladoD1RYDzhZo3AP9Aoce1hbaIZs34CGcN80o1yjWKJ43iCfbIRowLfCUfu33lcOmNmIlI6hMwAejcIbTzeiw71ycV5Ky2ClIIf5cymrYd052J5ep-2Z5IAMMCxCOrv4IGgoKCAgBEgTjYZkyDA&ved=0ahUKEwjNlZjI6rjKAhVEsIMKHX\_xD48Qwg4IGSgA&biw=1920&bih=935
* hiltTex.jpg
* rockTex.jpg
* pillarTex.jpg - https://www.google.ca/search?tbs=sbi:AMhZZiumfncyS5Jcu3UHZ58-pLOns6TDPMXAqPp8Jao1-xGzV0syRJgTVCrsJGGdtwv0g4HO9yz1uNHKwupWy1WJMioifxk3gsZ\_1cHm5wJG0NwGFYVPUk-FcpRydY5wBx2E1pWM41jewXxcLMImvWsKfksfiW-G1hm6WiiNjBLsjE8nR3kNbk4qdIHDFCGTFC79CbX6TnGyLdBgQPw0iuCqKg2IbCSi1v012PgRY-FV-4cDsZJXvx1MNgu3MOutph96LldOQ1U\_105maOCb8Z8MY30HMvTNMDzp6XPGwKmZySutlYrpu-v0DZb-4yvJFeVb1MKONCHf0zicDIS5yThJ2issduhU91JP6jXKHJkzaDKvNzzcCM9FVf7NuuYia3bDZiedQiYzvVeu7GMCSFu4KpHz7r2MTde8drPWhhUpVmTXNGNGabmDmuDhN95sDgyPt\_13yc11NPnZxRhW4DoAoztPNoYP\_1O0bn36bYGzrEjhf10mtBWANGdT6F\_1DVSXC2F9sqntzygpiiuvHuTbS9WjiNdxgqtqKjrRUoINPIdT8h0u4\_1bot6R4guVau--nPVkC0UC6ei\_1wSsqd\_15wEZUive69gIhaqt8O6dNlq8G0oWspvlJZsXOGCqfpDTBBurh\_1bbYrG1CI-gLlZKHXRYJoIQS5tR54OWFmCtVk9HsVzhXwAIYxlcQQlx3MOz\_1P-iJsN0Ue2ZGM\_1I0CtILlHGWWBms-nMwnqOT623Vcb-N\_1SPo1YFLa5nkEyqRz21tytlLJP25xzsMv6ETnkx1yRrAGVgwpnznyw\_10KoIx6Vpu42SdcCDb0kNOoCquMIKhFe69hRx18NqqP0D8typN5vaGnimtqRyCI9hxOf8h-wNii3D6bVwTYkgyM5W0tZRDDO\_1KoervFLQOaETS-Cs5ORv4KyUPCGdskYHUyb6lpE3IBKuxiUD9gnwlsj3ha3yQZ\_156mu3ts-LRRNrmSXql0HunG4Pl5PMtwsi-3\_1uhjBanA7VXHmC6K5fxmmutgkbJTTPfjUf8dCVVcCLkly4u2\_1QIWYOmZvfc5A9nPPKE96MwKXNSJthCciirWn6KtNEF2amK25\_1nr13nZfYv70Hc2swqztFLbbDPH\_1-snhP9LxqYPyPD5ngAmhBr5uDBmGx0cmGrfqP\_1nfsZYke5NiUaYXe5pbvXnAs1p8cI22V8fEuK9UTnhpoyf0XSNpL1cEbSu3L2Lr6EQbg86eflIeb9jEZBxueGiFZGm2SEo\_19IAw4WMMGV1FRBcAD9XWYFKrpe5GlHTjSRBwnXNQigllhKr0xMKAx8BBBn\_1wCJ6HUxRCP5XvjRH3jiGMm6kpy1tKwBnT-6GRd3oPbtTQ2yBz2n0C80UQWW87fRSUs8Uh-zRoepW9lGM1J-w7zUtvy8hubOSRgFAYdzm5c4C7fIdjc1TZ5\_1hR14ym08MDyhxU8F-bnWYPJRFtxcPqX1m4gvoMljl-UMwriBT5j-1QkiBMVyW-X2FcYsx71csMnb1WAxqXMHN2Z7flO7YEXy3M&btnG=Search%20by%20image&hl=en-CA
* logTex.jpg - https://www.google.ca/search?sa=G&hl=en-CA&q=old+wood+texture&tbm=isch&tbs=simg:CAQSiwEaiAELEKjU2AQaAggCDAsQsIynCBphCl8IAxInyAqhFaIVoBWjFQLHCqwVshXZB94p5SfpN-o31ijqJ9co6SfVPtM-GjClGxcRUb3aLWf6HeIpih7Sgr0SVrE8f1LqOGvThQNRgS98VlToSLAevbp7KWqqhbcgAwwLEI6u\_1ggaCgoICAESBJr-at8M&ved=0ahUKEwiZjb6y67jKAhUEkYMKHfr7DpIQwg4IGSgA&biw=1920&bih=935#imgrc=moEX7dSYytqSLM%3A
* floorTex.jpg - <https://www.google.ca/search?sa=G&hl=en-CA&q=sci+fi+ground&tbm=isch&tbs=simg:CAQSjgEaiwELEKjU2AQaBAgACAMMCxCwjKcIGmIKYAgDEijIFckVgx6CHoAewxXUC4EexxWsFek36jfuN9E-8TfrN_1I30z7fKdI-GjAOkBc9AlLbCwmYF8DluLQ0H7ctVvJlGB77zTlNBoB_1tFels4bNP4NNCV6OQiO4LXMgAwwLEI6u_1ggaCgoICAESBANdnmMM&ved=0ahUKEwjbi72I67jKAhXMn4MKHQudDiMQwg4IGSgA&biw=1920&bih=935#imgrc=u-Bc1INmxHihKM%3A>
* treeTex.jpg - https://www.google.ca/search?tbs=sbi:AMhZZiteifMu9erqp989STOBdvC1BI39QffaGFM0zgPkppcPkgZRTs0zLIHPJA\_1QoR4x-MMEP1MEPMwPAviXR0mIvoxHqJzSLCKAwYWsDOoEaYqsUsVaijnDO6xUVHxjJeQuJORNGu8ylx9C99r9KJ3WUiA3Q0OCX-TwHDinw4UmTwr\_1xAuC7cQ4tkwpN9QV3-wWZ6i2b1WEMJdkXZLnTRwbx0mF1GM07W2VZvdALdoaG2i-iLlFzjHv6mlLssuvvQJBZetNsEsCwgwhAMiOYHmu3JuUfYE-mCxYgB0l0YmaCxFikZQoqFM7R8YHBUMsTUF3MU1On-z2gxiWY4iB8ftTGH40Q2PYJiDl5-m6NyQid1\_1EPi7jiX0fxshYiadTGXCub-PUyI\_1h8dgGbHUEPUSu5doPCOFUWgUNWpOcxGCmjjFY7B5eTLldQKvqWb7zfDem6WOu3oKPZCVWWJTvAMVpLJ-IeEmnzCX4mYzjaShiXuEjrk0FZSG9vohLw-BCZ4-g-5ZFSj\_1JjPPPGelwDrYQdADqn3n6a7v-vrqBrFnY2r1jpd6MS3\_1VZZ3gwNOBdYj5bKfGMwoFDVrcli6s2AI2fcjT0xN4OdZAuBsTFCS8f9ucm3XlAw8zjwsn6j62W9jN0OEZarKuWIZPM79PMlSZN1g\_1o0mDzKs0y\_1UsxcMxwFvB4PwAIEAqQ8Jj889WKwGLwaIxOhnhdPCezZckLpecIvClHtr\_1Q1wOImi8k1cNLJ0Cg0ywXYeHI\_1U5sNHSpVu9thSQXf8eM6FM3yP0fkqCjJzNoHvJcKs1rW-G-dwrqs8NJN0knO4mwNBkM5vhUFO0kUC82AOJwIK2\_1ElEhA7XqWGOta7DoB6L3r4BSiA7tDe-v\_1BGs8zTr64WIcfbJp4axMI95nfmAzjLXe\_1OBsa\_18RNC0vs4VfDyR8lP03v373kz8nULI0nNYTAjJ0Ph7vjOhHGcVl\_1Xw43rC3QHVyZbScCScw-dGn6OCd5xmxV21dLLj-1gBC17wpI-G0xXlSdznItr552eMZMJxZ-a2eDPBiFbqxKmNKGW2eZi0eodqU3pWozgGwQ-Zqq2DZNCKQctyZkj1BnF8zWo0nDxVz0Yju\_1DpBnLDZJNikg7OyCZWMflFJFL1avckF0EyIQrpFIUvstRDGvt1sdLFxI-eJs6yE7HHd4PvMx79a0BqkiR\_1QSs\_1pbQ737CB2n9Yz8uNGUliNbzZ03qI1rHOLbMyvJ5uKo2gvmFLgBPX5dhex7Epo3FWjzmDnrmR-gYTvJeVO5XnE6yVroJ1Q7d3SSEYncd\_1\_1gWmCZ3e6h761oOlqV5JfayJrneJ4qoLi7qmyDjwd54Quoy5FAuiJFFKhEWaeJqqw8LiGhhb50LI0YIMysQkOIyyBwi\_1ice-oOqvMIuzgpTrfuNGg0Bf0KjJTO7NpDQgpXeeFBYiEpaeYlCI0ziI-gUJG1ryZv6kldQzYt23pi8xdmiuVCO0en-ewOJzwxdH49uYwH0YQ4aN0zM8VbwVN43jGYlZ8mKhrRXLnilAgRHTqnA6ztv&btnG=Search%20by%20image&hl=en-CA
* rockTex.jpg - https://www.google.ca/search?tbs=sbi:AMhZZiteifMu9erqp989STOBdvC1BI39QffaGFM0zgPkppcPkgZRTs0zLIHPJA\_1QoR4x-MMEP1MEPMwPAviXR0mIvoxHqJzSLCKAwYWsDOoEaYqsUsVaijnDO6xUVHxjJeQuJORNGu8ylx9C99r9KJ3WUiA3Q0OCX-TwHDinw4UmTwr\_1xAuC7cQ4tkwpN9QV3-wWZ6i2b1WEMJdkXZLnTRwbx0mF1GM07W2VZvdALdoaG2i-iLlFzjHv6mlLssuvvQJBZetNsEsCwgwhAMiOYHmu3JuUfYE-mCxYgB0l0YmaCxFikZQoqFM7R8YHBUMsTUF3MU1On-z2gxiWY4iB8ftTGH40Q2PYJiDl5-m6NyQid1\_1EPi7jiX0fxshYiadTGXCub-PUyI\_1h8dgGbHUEPUSu5doPCOFUWgUNWpOcxGCmjjFY7B5eTLldQKvqWb7zfDem6WOu3oKPZCVWWJTvAMVpLJ-IeEmnzCX4mYzjaShiXuEjrk0FZSG9vohLw-BCZ4-g-5ZFSj\_1JjPPPGelwDrYQdADqn3n6a7v-vrqBrFnY2r1jpd6MS3\_1VZZ3gwNOBdYj5bKfGMwoFDVrcli6s2AI2fcjT0xN4OdZAuBsTFCS8f9ucm3XlAw8zjwsn6j62W9jN0OEZarKuWIZPM79PMlSZN1g\_1o0mDzKs0y\_1UsxcMxwFvB4PwAIEAqQ8Jj889WKwGLwaIxOhnhdPCezZckLpecIvClHtr\_1Q1wOImi8k1cNLJ0Cg0ywXYeHI\_1U5sNHSpVu9thSQXf8eM6FM3yP0fkqCjJzNoHvJcKs1rW-G-dwrqs8NJN0knO4mwNBkM5vhUFO0kUC82AOJwIK2\_1ElEhA7XqWGOta7DoB6L3r4BSiA7tDe-v\_1BGs8zTr64WIcfbJp4axMI95nfmAzjLXe\_1OBsa\_18RNC0vs4VfDyR8lP03v373kz8nULI0nNYTAjJ0Ph7vjOhHGcVl\_1Xw43rC3QHVyZbScCScw-dGn6OCd5xmxV21dLLj-1gBC17wpI-G0xXlSdznItr552eMZMJxZ-a2eDPBiFbqxKmNKGW2eZi0eodqU3pWozgGwQ-Zqq2DZNCKQctyZkj1BnF8zWo0nDxVz0Yju\_1DpBnLDZJNikg7OyCZWMflFJFL1avckF0EyIQrpFIUvstRDGvt1sdLFxI-eJs6yE7HHd4PvMx79a0BqkiR\_1QSs\_1pbQ737CB2n9Yz8uNGUliNbzZ03qI1rHOLbMyvJ5uKo2gvmFLgBPX5dhex7Epo3FWjzmDnrmR-gYTvJeVO5XnE6yVroJ1Q7d3SSEYncd\_1\_1gWmCZ3e6h761oOlqV5JfayJrneJ4qoLi7qmyDjwd54Quoy5FAuiJFFKhEWaeJqqw8LiGhhb50LI0YIMysQkOIyyBwi\_1ice-oOqvMIuzgpTrfuNGg0Bf0KjJTO7NpDQgpXeeFBYiEpaeYlCI0ziI-gUJG1ryZv6kldQzYt23pi8xdmiuVCO0en-ewOJzwxdH49uYwH0YQ4aN0zM8VbwVN43jGYlZ8mKhrRXLnilAgRHTqnA6ztv&btnG=Search%20by%20image&hl=en-CA
* lightningTex.jpg - <https://www.google.ca/search?tbs=sbi:AMhZZit0eEyYaMEf82JEvcYbtBS9LPoZUzruubVq6XJbS167jQ5U5fot6MytmSoW8p1TYzqj0Y2k1ndMbadwc-qzpwVcWUTRcHMnvBu-7XMkcMgTPgcwloYs7pO24UdjLBKjggQf6pn6QC4zhzkEkgC5ttW5ECOc95al6EeY84W7YuA-wYwsAj7NuyGNeK_1R9aFO3BTQZ8JH0ncxUjPTmLH_1TgciMxayI7IGzwswy4RxXxElEY_1PUj-83yz9obdJkn45UwO8fcOBvSMwAPXsJv4KNTfmQQjKMg0YlZrcU60ysp3w_1eeZDDqLrB3wO4KR9ELABLyQY6pSHg8YfHBNu6F9dw1rok6bK501unEcP_1HMEZp5NXXUsBDDiEV-thkOwvLnqc95WVS8oZN5CLFBF6_1R6hv4nV3Ef7WOlyaEU7opo0CQSxceUXA7bB4alBbBQNcK9gMe08hXR4svqYDoMFo5p1G-8EYO_1w&btnG=Search%20by%20image&hl=en-CA>

Audio

The Sundering by The Sword

Hail and Kill by Manowar