Testing Plans for each stage of Project-B:

Stage 1:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front right corner, halfway up, and looks straight toward the opposite corner of the environment. The user is unable to move the viewpoint unless the program is re-executed with different starting values.

Bad Data:

1) Surface offset:
Each surface of the room is offset by a factor of 0.1. This is to prevent stitching from occuring with the grid lines in the room. When viewed up close, the seams are visible. However, the presense of the lines help mitigate the unpleasant gap effect. This looks much better than the stitching of the grid lines if no offset were used.

2) Extra lines:
When creating the grid lines, there are double lines produced along the edges where surfaces meet. This is because each surface is created in pairs. The overlap is not visible to the user. Additionally, the extra 8 lines will probably not have much computational effect as they are quite easy to render. Therefore, I have let them stand, as I felt it was not worth the extra programming effort to remove them.

Stage 2:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front right corner, halfway up, and looks straight toward the opposite corner of the environment. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is textured to appear as an eyeball facing the front wall.

Bad Data:

1) Character position:
The character position is set assuming that there is no offset in any of the surfaces. The gridlines represent the actual surface of the environment. I have ignored the offset because it should be small enough to not be noticed unless the view is placed extremely close to the object.

Stage 3:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall.

Bad Data:

1) Other views:
If the position of the initial view is changed to another wall, the point where the user needs to look to maintain a 45 degree angle needs to be calculated by hand. Since the user can't move the view at this point, and future moves are made via mouse, I felt that adding in this functionality was unecessary.

Stage 4:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can now be used to control the camera in this scene. A left button drag will change the rotation of the camera. A right button drag will change the camera position along the x and y axes. Using the mouse will will allow the camera to zoom in and out along the z axis. The camera position is maintained within the boundaries of the room structure.

Bad Data:

1) Camera position:
The camera position is checked each frame by overriding the orbit control methods. The center of rotation is updated to the current camera position so that the rotation occurs around the camera itself. The position is checked and adjusted to keep the camera position within the bounds of the room structure. However, when I set the check values to be exactly the edges of the room and and adjusted the values to be the edges of the room also, the camera would sometimes get stuck. This is why the check values for the position of the camera have a slight offset. This prevents the camera from getting stuck, and the offset is small enough such that it should not be detectable by the user.

Stage 5:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys outlined in the project for this stage.

Bad Data:

1) User tries to move character outside room bounds:
The character position is kept within the bounds of the room with a check against each movement. The room bounds are the values for the grid positions, as the actual room structure is slightly offset.

2) A key is pressed other than specified for this stage:
The user is free to press any key, but only the specified keys will respond to input. Any other key press at any other time does not have any effect on the virtual scene.

Stage 6:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys outlined in the project for stage 5. An textured cube is placed in the scene and acts as an obstacle. As the character moves, it will not be allowed to pass through the obstacle.

The requirements for this stage also state that character stop moving when touching a ball. However, the ball creation step is outlined in the next stage. I will implement this functionality at that point.

Bad Data:

1) User tries to move over area covered by the obstacle object:
The character is prevented from colliding with the obstacle object through a check of the

intersection of the collision bounds. The move will not take place if there is an intersection. All other moves within the room environment will be allowed as in the previous stage.

2) No touching edges:
Because the character moves in increments 0.1 units through the scene, it appears that the object stops before coming into contact with the obstacle or walls. However, if the dimensions of the obstacle or character are changed, this gap will be reduced or disappear entirely.

3) Obstacle creation:
When the obstacle is created, the maximum size allowed is the space left in the room minus the size of the central character. If the user tries to exceed this amount, the default size of the box is 0.5, which creates a 1x1x1 box. If the user attempts to add the object without setting the position, the obstacle will default to a position of (1,1) in the xz plane. Whether the position is set or not, the position values are checked to make sure that the object created will be positioned within the bounds of the room and will not overlap the central character. If the position is outside the room, the same default value is attempted as above. If any of the above positions will overlap the central character, the x and z values are adjusted such that the object fits just beside the central character.

Stage 7:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys outlined in the project for stage 5. An textured cube is placed in the scene and acts as an obstacle. As the character moves, it will not be allowed to pass through the obstacle. A small textured sphere is placed in the scene which is moved by making the character come into contact with the ball. When contact occurs, the ball stops and the ball continues to move in the direction that the character moved to hit it. The ball itself will stop when coming into contact with the walls of the room or the obstacle object. If no contact occurs, the ball moves at a constant rate for a set period of time.

Bad Data:

1) No touching edges:
As in the previoius stage, all objects do not quite touch each other when coming into contact. This is because the collision detection merely stops motion when a collision is detected. The move increments of the ball may not allow it to come into perfect contact with the character or obstacle.

2) Ball creation:

The ball is created to be smaller than the character and placed at an initial starting position of (x, z) = (1, 1.8). This placement does not detect if there is currently an obstacle or character occupying that postion. Since I am creating this scene by hand, there is no reason to automatically check. The ball size and position can be changed if necessary, by modifying the initial values within the Ball class.

Stage 8:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys outlined in the project for stage 5. An textured cube is placed in the scene and acts as an obstacle. As the character moves, it will not be allowed to pass through the obstacle. A small textured sphere is placed in the scene which is moved by making the character come into contact with the ball. When contact occurs, the ball stops and the ball continues to move in the direction that the character moved to hit it. The ball itself will stop when coming into contact with the walls of the room or the obstacle object. If no contact occurs, the ball moves at a constant rate for a set period of time. Both the ball and character will rotate based on the direction of movement. The character faces the direction of movement and the ball rotates as if rolling in the direction of movement.

Bad Data:

1) Character rotation:

The character rotates in the direction of movement based on the key pressed. This rotation occurs no matter what the result of the movement. Thus, the character always faces the direction of the last valid key press, even if no move occurs because of collision.

Stage 9:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys

outlined in the project for stage 5. An textured cube is placed in the scene and acts as an obstacle. As the character moves, it will not be allowed to pass through the obstacle. A small textured sphere is placed in the scene which is moved by making the character come into contact with the ball. When contact occurs, the ball stops and the ball continues to move in the direction that the character moved to hit it. The ball itself will stop when coming into contact with the walls of the room or the obstacle object. If no contact occurs, the ball moves at a constant rate for a set period of time. Both the ball and character will rotate based on the direction of movement. The character faces the direction of movement and the ball rotates as if rolling in the direction of movement. With the character jump, the jump goes to an initial height that is 2x the height of the character. It moves up at a rate of 1.5 and down at a rate of 0.98. Each successive jump is 90% as high as the previous jump. Once the height is less than 10% of the original jump, the animation ends.

Bad Data:

1) User attempts to move character during jump animation.
   Once the jump is initiated, the character x/z position is locked, as well as the rotation. The character does not respond to any key press until the animation is finished.

Stage 10:

Normal case:

A 10x10x10 3D environment is created with a grassy floor, stone walls, and cloudy blue sky. Each surface is divided into a 20x20 grid using yellow lines. The initial viewpoint is set to the front wall, 1/3 of the way up, and looks downward at a 45 degree angle. The user is unable to move the viewpoint unless the program is re-executed with different starting values. A textured sphere representing the character is in the center of the room on the floor surface. It is visible in the center of the floor facing the front wall. The mouse can be used to control the camera in this scene. The central character can be controlled by pressing the specified keys outlined in the project for stage 5. An textured cube is placed in the scene and acts as an obstacle. As the character moves, it will not be allowed to pass through the obstacle. A small textured sphere is placed in the scene which is moved by making the character come into contact with the ball. When contact occurs, the ball stops and the ball continues to move in the direction that the character moved to hit it. The ball itself will stop when coming into contact with the walls of the room or the obstacle object. If no contact occurs, the ball moves at a constant rate for a set period of time. Both the ball and character will rotate based on the direction of movement. The character faces the direction of movement and the ball rotates as if rolling in the direction of movement. With the character jump, the jump goes to an initial height that is 2x the height of the character. It moves up at a rate of 1.5 and down at a rate of 0.98. Each successive jump is 90% as high as the previous jump. Once the height is less than 10% of the original jump, the animation ends.

Bad Data:

1) Jumping animation:
   The jump motion is as specified in the requirements for this stage, however, I do not allow

the user to move the character while the animation plays.

2) Audio:

The background audio is played when the main() method is called to execute the program. It is not actually contained in any of the classes for the scene. Additionally, the audio for jumping/bouncing can cause a delay in the running of the program when playing for the first time.

3) Rain:

The falling rain animation makes splashes when coming into contact with the awning/floor. This animation scales the shape until it reaches a certain size and then disappears. However, at the extreme points of the room/awning, this animation will cause the splash shape to exceed the bounds of the awning/room. I feel that the splashes animate quickly enough that this effect is not very noticable to the user.