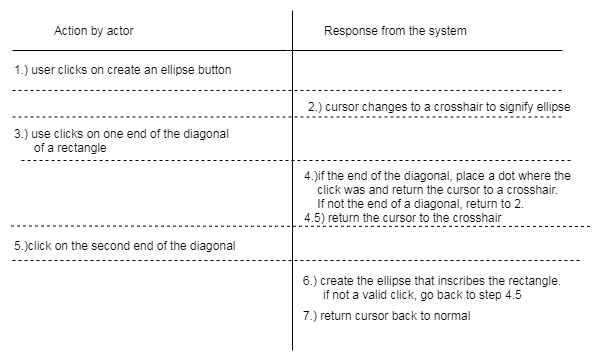
**Create Ellipse:**

****

Step 1 is to be carried out by the controller, to gain input from the user selection to create an ellipse.

Step 2 is to be carried out by the view, which changes the cursor to a crosshair and changes the view for the user.

Step 3 is to be carried out by the controller, since it is waiting for an input from the user (a click at one end of the diagonal).

Step 4 is to be carried out by the model, since it stores the data of the location of the first click

Step 4.5 is to be carried out by view since it just changes the cursor back to a crosshair

Step 5 is to be carried out by the controller since it is again waiting for user input.

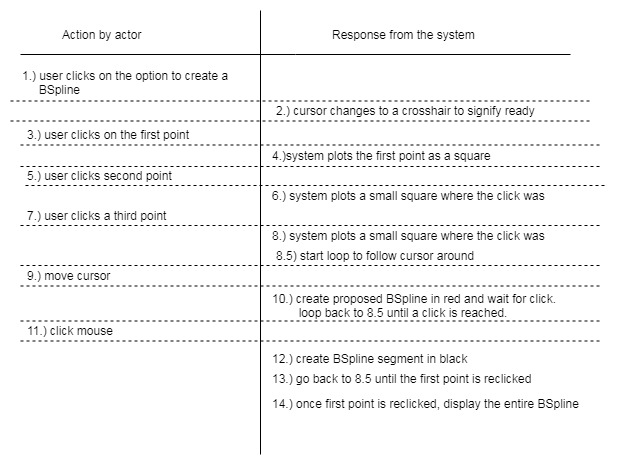
Step 6 is to be carried out by the model since it is creating and storing the ellipse.

Step 7 is to be carried by the view since it is just returning the cursor back to normal.

**Flow of Ellipse Method:**

Switch to ellipse mode by waiting for the user to click the create ellipse button. This invokes a method on the controller called CreateEllipse() and is triggered when the ellipse button is pressed. Next, the view changes the cursor to a crosshair with a method called changeCursorToCrosshair(). The user clicks the mouse, which invokes a method on the controller called EllipseFirstClick() and returns the coordinates of Click1. The model stores the location of the first click with StorePoint(Click1) method. The cursor is then changed back to a crosshair, carried out by the changeCursorToCrosshair() method in view. Then the user clicks the mouse again, which invokes a method in the controller called EllipseSecondClick() and returns the coordinates of Click2. The model then creates the ellipse by calling CreateEllipse(Click1, Click2) and stores and displays the ellipse.

**Create BSpline:**

****

Step 1 is to be carried out by the controller, to gain input from the user selection to create BSpline

Step 2 is to be carried out by the view, which changes the cursor to a crosshair and changes the view for the user.

Step 3 is to be carried out by the controller, since it is waiting for an input from the user

Step 4 is to be carried out by the model, since it stores the data of the location of the first click

Step 5 is to be carried out by the controller, since it is waiting for an input from the user

Step 6 is to be carried out by the model, since it stores the data of the location of the second click

Step 7 is to be carried out by the controller, since it is waiting for an input from the user

Step 8 is to be carried out by the model, since it stores the data of the location of the third click

Step 8.5. Is to be carried out by the controller since it is getting input from where the mouse is moving

Step 9 is to be carried out by the controller since it is following the cursor around

Step 10 is to be carried by the controller and view since it is moving the proposed red BSpline around (which changes view) and following mouse (which is handled by controller)

Step 11 is handled by the controller since it is waiting for a user click

Step 12 is to be handled by the model since it is creating a segment of the BSpline

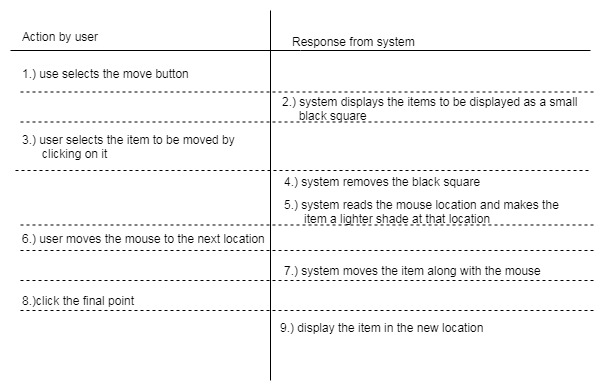
Step 13 is to be handled by the controller since it is waiting for the first point to be re-clicked

Step 14 is to be handled by the model since it creates and stores the completed BSpline

**Flow of BSpline:**

Switch to Bspline mode by waiting for the user to click the Bspline button. This invokes a method on the controller called CreateBSpline() and is triggered when the BSpline button is pressed. Next the cursor changes to a crosshair with a method called changeCursorToCrosshair(). The user clicks the mouse at a desired first location, which invokes a method called BSplineClick1() and returns the coordinates of click 1. A square is placed at the first click by invoking a method on view called CreateSquare(Click1) and the model stores the coordinates of click 1 with a method in the controller called StorePoint(Click1). The user clicks the mouse at a desired second location, which invokes a method called BSplineClick2() and returns the coordinates of click 2. A square is placed at the second click by invoking a method on view called CreateSquare(Click2) and the model stores the coordinates of click 2 with a method in the controller called StorePoint(Click2). The user clicks the mouse at a desired third location, which invokes a method called BSplineClick3() and returns the coordinates of click 3. A square is placed at the first click by invoking a method on view called CreateSquare(Click3) and the model stores the coordinates of click 3 with a method in the controller called StorePoint(Click3). A loop is started in a method called FinishBSpline(), to loop until the first point is reclicked. Then the mouse is followed around with a method in the controller called FollowMouse(), which follows the mouse location around in a loop and invokes a method in view called PossibleBSpline() each time the mouse is moved to update and display a possible BSpline in red. Once another point is clicked, a method in controller is invoked called StorePoint() which stores the point, and invokes a method in the model called CreateSpline() which creates and stores the new BSpline segment. The loop in FinishBSpline() is restarted, until point 1 is reclicked. A method in the model is then invoked, called StoreBSpline(), which stores and displays the finished BSpline.

**Move Item:**

****

Step 1 is to be carried out by the controller, to gain input from the user selection to move an item

Step 2 is to be carried out by the view, which displays possible movable items with a black square

Step 3 is to be handled by the controller since it is waiting for a selection from the user

Step 4 is to be carried out by the view since it just simply removes the black square

Step 5 is to be carried out by the view and the controller since it follows the mouse around (handled by controller) and moves the item around in a lighter shade with the mouse (changing the view).

Step 6 is handled by the controller since it is following the mouse movement

Step 7 is to be carried out by the view since it is moving the item around with the mouse and displaying it

Step 8 is to be done by the controller since it is waiting for a final click

Step 9 is to be handled by the model since it stores the new location of the item

**Flow of Move Item:**

Switch to move item mode by waiting for the user to click on the Move Item button. This invokes a method on the Controller called MoveItem() and is triggered when the Move Item button is pressed. MoveItem() invokes a method right away called HighlightItems() in view, which displays black squares by items that can be moved. The controller than waits for a user to click and select a movable item, which is done with a method called SelectItem(Click) in controller. The view then removes the black square with a method called RemoveHighlight(), and then a method in the controller is invoked called followMouse() that follows the mouse around. Within follow mouse, a method called DragItem() is invoked in view, which moves the item around with the mouse in a lighter color shade. The user then clicks in a new location, which is done by a method called PlaceItem(Click2) in controller. This method invokes a method in the model called StoreMove() to store the new item location.