Code critique by Georgy Khomchenko

Providers: Khoi Pham, Siddharth Singh

1. View panel interacts with model directly

Drawing component (JPanel) of VisualView takes model in a constructor and directly makes changes in it according to the current tick. After asking model to animate its shapes according to the current time, this visual component again calls the model directly and asks to provide shapes for drawing. I think this is a violation of MVC pattern where controller should be the only one component that is able to make changes to the model.

Proposed improvement:

From my prospective, it would much better for the view to take some sort of data that it can use to draw shapes on its own. For example, it could take both shapes and animations and then apply animations to the shapes according to current time. In that case it would be independent of the model and would be able to produce shapes needed to draw on its own. Another possible solution would be to create frames, where each frame would represent an array of shapes with all parameters being set according to specific point of time. As an example, if you had 10 ticks long animation involving 2 shapes, you would have 10 frames each containing array of 2 shapes. This frame approach makes it way easier to adopt model to different sort of views.

1. For actions that require additional parameters, controller was responsible for getting these parameters

For all buttons edit view sets controller as action listener. Even though it is fine to delegate simple function like start or pause directly to controller, functionality that require some additional parameters like addShape or modifyKeyFrame, should first make user input these parameters and only then ask controller to proceed with request. From what I received from providers I can judge that controller was responsible for displaying window to the user that will ask to enter parameters.

Proposed improvement:

Transfer this parsing behavior to the edit view, so when buttons that would require parameters are pressed, edit view will first display text fields to the user to read parameters from them and only then will call controller to apply modifications.

1. Mutable shapes instead of set of copies of the starting shapes

Based on how drawing panel behaves (after calling **model.motionAnimate(animation, time)** it calls **model.getShapes()** to get shapes that needs to draw right now) I can assume that motionAnimate(animation, time) would mutate shapes to correspond to the current point of time. It makes sense if we assume that animation is only going forward, but it would be pretty hard to play backwards. And even if you would make it work, you will simply recompute all animation but starting from the end. And besides that, mutation is not the safest approach to use which theoretically can cause problems when adding features to your code.

Proposed improvement:

Existing method **motionAnimate(animation, time)** instead of mutating starting shapes it could create copies with modified fields (position, size, color) and store them. In this case by the end of animation you would have all forms of shapes throughout the animation. It would make you model more flexible for possible features and make it be more adaptable to different views/controllers.

1. Operating with Abstract class instead of interface in some signatures

In providers model, In methods related to shapes (addShape, getShapes) providers operate with AShape (abstract class) instead of IShape (shape interface). It is always a better practice to operate with interfaces, since all public functionality should be declared there. Proposed solution is fairly simple – try to stick to interface in method declarations

1. Public method addShape (AShape shape) requires instance of AShape to be passed in

Public method AddShape requires AShape as a parameter to be passed in, which means that controller is responsible to construct instance of AShape and then pass it in to the model. It doesn’t look right to delegate construction of a shape to the controller, since it is a model’s responsibility.

Proposed improvement:

Make this method take only name and the type of the shape (or also position, color and size), in that case controller will simply pass along these parameters and won’t be overloaded with shape construction that should definitely happen inside of model.

1. Controller wasn’t hidden behind some listener interface
2. Mutable shapes, would be hard to make backwards animation
3. Operating with abstract class AShape instead of IShape interface
4. Uncertain method in model interface, contradicts with motion animate