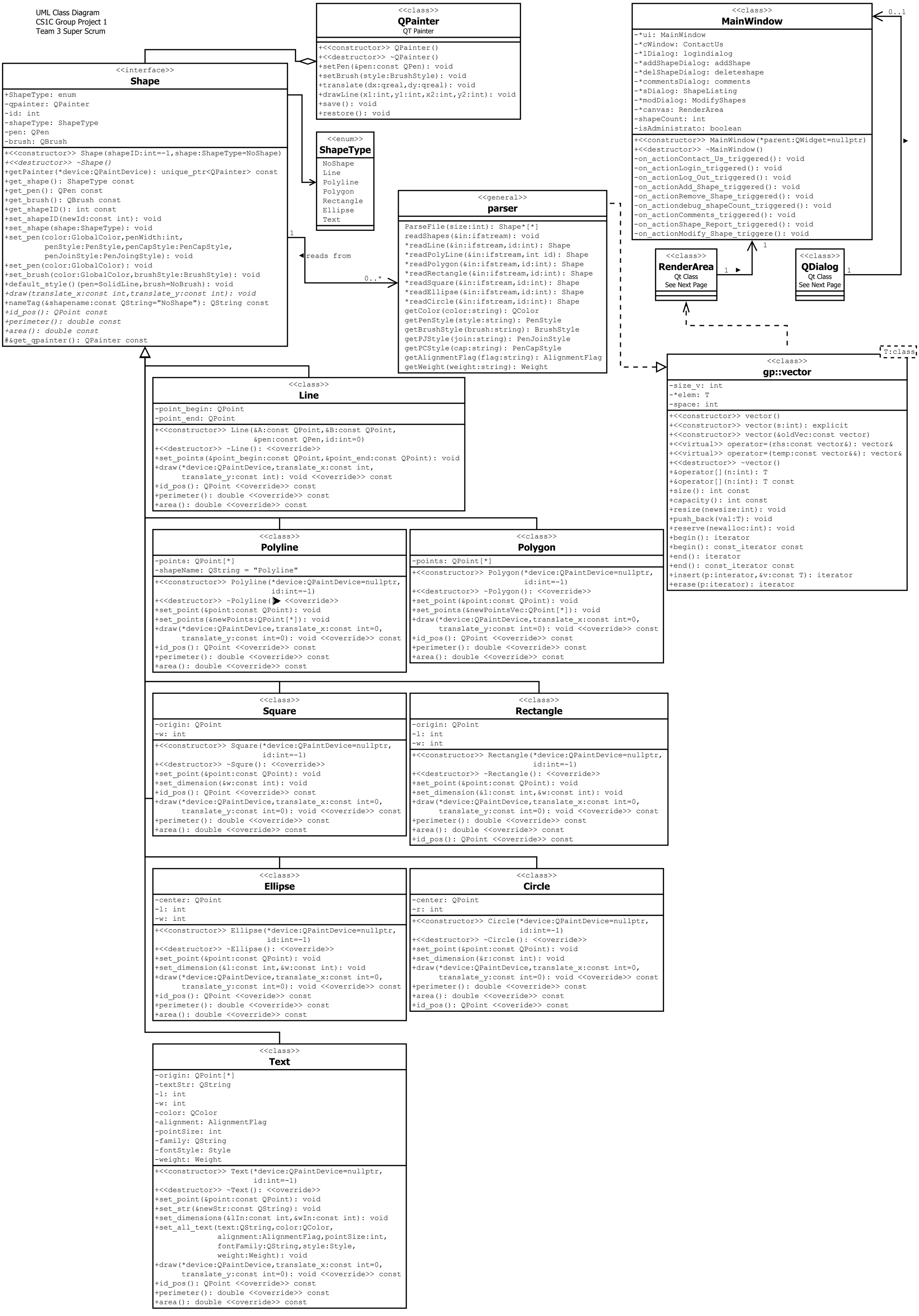
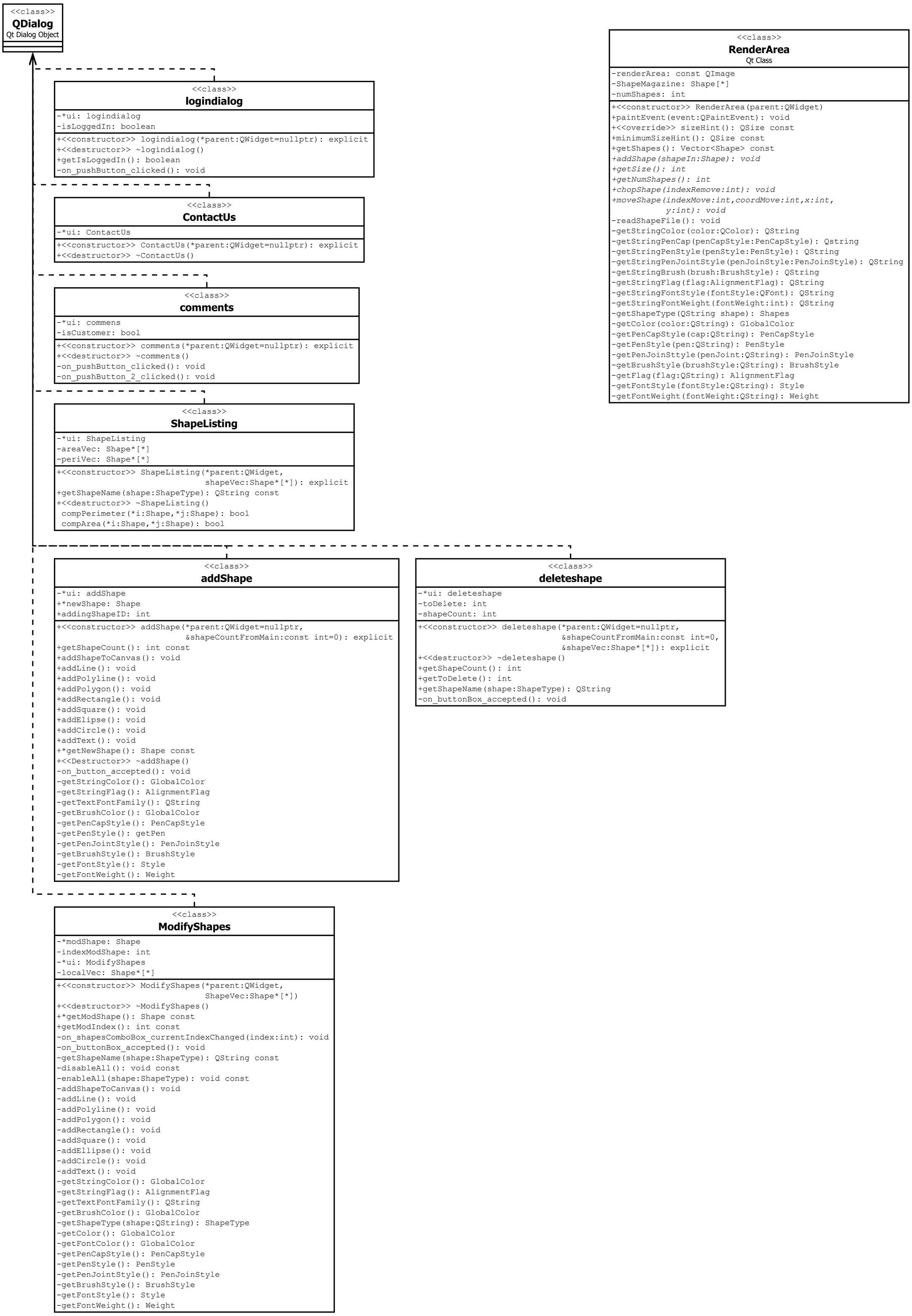
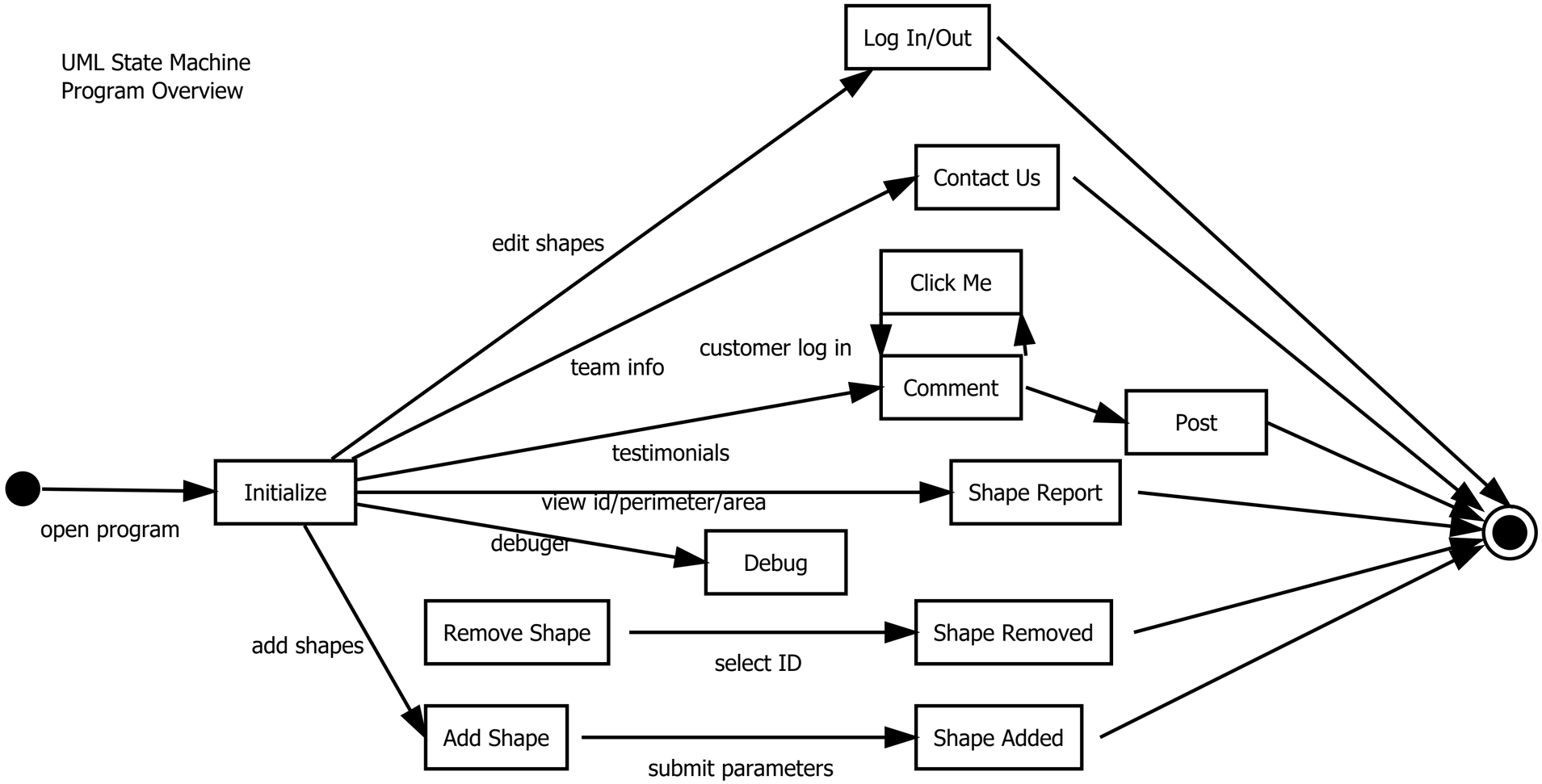
UML Class Diagram
CS1C Group Project 1
Team 3 Super Scrum




<<class>>
QDialog
Qt Dialog Object
<<class>>
logindialog
-*ui: logindialog
-isLoggedIn: boolean
+<<constructor>> logindialog(*parent:QWidget=nullptr): explicit
+<<destructor>> ~logindialog()
+getIsLoggedIn(): boolean
-on_pushButton_clicked(): void
<<class>>
ContactUs
-*ui: ContactUs
+<<constructor>> ContactUs(*parent:QWidget=nullptr): explicit
+<<destructor>> ~ContactUs()
<<class>>
comments
-*ui: commens
-isCustomer: bool
+<<constructor>> comments(*parent:QWidget=nullptr): explicit
+<<destructor>> ~comments()
-on_pushButton_clicked(): void
-on_pushButton_2_clicked(): void
<<class>>
ShapeListing
-*ui: ShapeListing
-areaVec: Shape*[*]
-periVec: Shape*[*]
+<<constructor>> ShapeListing(*parent:QWidget,
shapeVec:Shape*[*]): explicit
+getShapeName(shape:ShapeType): QString const
+<<destructor>> ~ShapeListing()
compPerimeter(*i:Shape,*j:Shape): bool
compArea(*i:Shape,*j:Shape): bool
<<class>>
addShape
-*ui: addShape
+*newShape: Shape
+addingShapeID: int
+<<constructor>> addShape(*parent:QWidget=nullptr,
&shapeCountFromMain:const int=0): explicit
+getShapeCount(): int const
+addShapeToCanvas(): void
+addLine(): void
+addPolyline(): void
+addPolygon(): void
+addRectangle(): void
+addSquare(): void
+addElipse(): void
+addCircle(): void
+addText(): void
+*getNewShape(): Shape const
+<<Destructor>> ~addShape()
-on_button_accepted(): void
-getStringColor(): GlobalColor
-getStringFlag(): AlignmentFlag
-getTextFontFamily(): QString
-getBrushColor(): GlobalColor
-getPenCapStyle(): PenCapStyle
-getPenStyle(): getPen
-getPenJointStyle(): PenJoinStyle
-getBrushStyle(): BrushStyle
-getFontStyle(): Style
-getFontWeight(): Weight
<<class>>
deleteshape
-*ui: deleteshape
-toDelete: int
-shapeCount: int
+<<constructor>> deleteshape(*parent:QWidget=nullptr,
&shapeCountFromMain:const int=0,
&shapeVec:Shape*[*]): explicit
+<<destructor>> ~deleteshape()
+getShapeCount(): int
+getToDelete(): int
+getShapeName(shape:ShapeType): QString
-on_buttonBox_accepted(): void
<<class>>
ModifyShapes
-*modShape: Shape
-indexModShape: int
-*ui: ModifyShapes
-localVec: Shape*[*]
+<<constructor>> ModifyShapes(*parent:QWidget,
ShapeVec:Shape*[*])
+<<destructor>> ~ModifyShapes()
+*getModShape(): Shape const
+getModIndex(): int const
-on_shapesComboBox_currentIndexChanged(index:int): void
-on_buttonBox_accepted(): void
-getShapeName(shape:ShapeType): QString const
-disableAll(): void const
-enableAll(shape:ShapeType): void const
-addShapeToCanvas(): void
-addLine(): void
-addPolyline(): void
-addPolygon(): void
-addRectangle(): void
-addSquare(): void
-addEllipse(): void
-addCircle(): void
-addText(): void
-getStringColor(): GlobalColor
-getStringFlag(): AlignmentFlag
-getTextFontFamily(): QString
-getBrushColor(): GlobalColor
-getShapeType(shape:QString): ShapeType
-getColor(): GlobalColor
-getFontColor(): GlobalColor
-getPenCapStyle(): PenCapStyle
-getPenStyle(): PenStyle
-getPenJointStyle(): PenJoinStyle
-getBrushStyle(): BrushStyle
-getFontStyle(): Style
-getFontWeight(): Weight
<<class>>
RenderArea
Qt Class
-renderArea: const QImage
-ShapeMagazine: Shape[*]
-numShapes: int
+<<constructor>> RenderArea(parent:QWidget)
+paintEvent(event:QPaintEvent): void
+<<override>> sizeHint(): QSize const
+minimumSizeHint(): QSize const
+getShapes(): Vector<Shape> const
+addShape(shapeIn:Shape): void
+getSize(): int
+getNumShapes(): int
+chopShape(indexRemove:int): void
+moveShape(indexMove:int,coordMove:int,x:int,
y:int): void
-readShapeFile(): void
-getStringColor(color:QColor): QString
-getStringPenCap(penCapStyle:PenCapStyle): Qstring
-getStringPenStyle(penStyle:PenStyle): QString
-getStringPenJointStyle(penJoinStyle:PenJoinStyle): QString
-getStringBrush(brush:BrushStyle): QString
-getStringFlag(flag:AlignmentFlag): QString
-getStringFontStyle(fontStyle:QFont): QString
-getStringFontWeight(fontWeight:int): QString
-getShapeType(QString shape): Shapes
-getColor(color:QString): GlobalColor
-getPenCapStyle(cap:QString): PenCapStyle
-getPenStyle(pen:QString): PenStyle
-getPenJoinSttyle(penJoint:QString): PenJoinStyle
-getBrushStyle(brushStyle:QString): BrushStyle
-getFlag(flag:QString): AlignmentFlag
-getFontStyle(fontStyle:QString): Style
-getFontWeight(fontWeight:QString): Weight

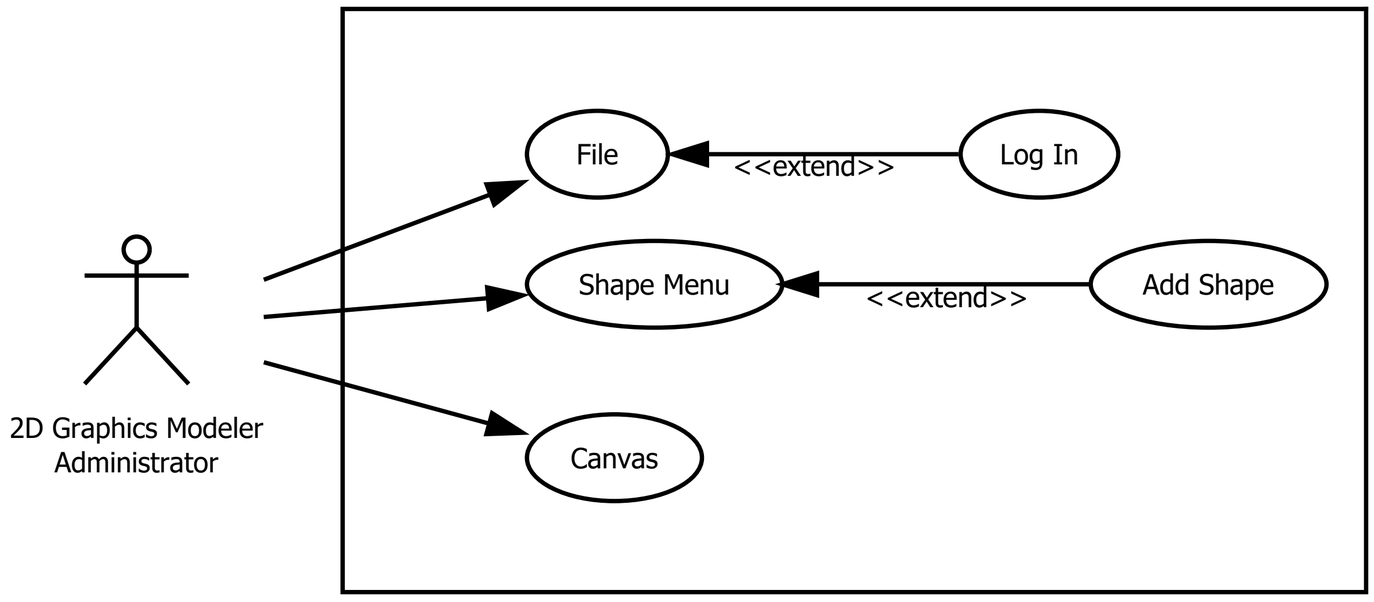
UML State Machine
Program Overview



UML Use Case 01 – Move Shapes

Use Case Number	01
Application	2D Graphics Modeler (Super Scrum)
Use Case Name	Move Shapes
Use Case Description	The actor can move shapes being rendered.
Primary Actor	2D Graphics Modeler Administrator
Precondition	• Initialize shape list • Displaying shapes on canvas • Actor is logged in as Administrator
Trigger	The actor moves shapes on canvas with their mouse.
Basic Flow	1. Actor navigates to the File tab in the menu bar 2. Actor selects Log In in the drop down menu 3. Actor logs into Administrator account 4. Actor navigates to the shape menu 5. Actor adds a shape to the canvas 6. Actor navigates to the canvas and moves rendered shape
Alternate Flows	• If shape already exists on canvas then Actor does not need to add a new one • Actor can create a shape or use an existing one from the shape menu

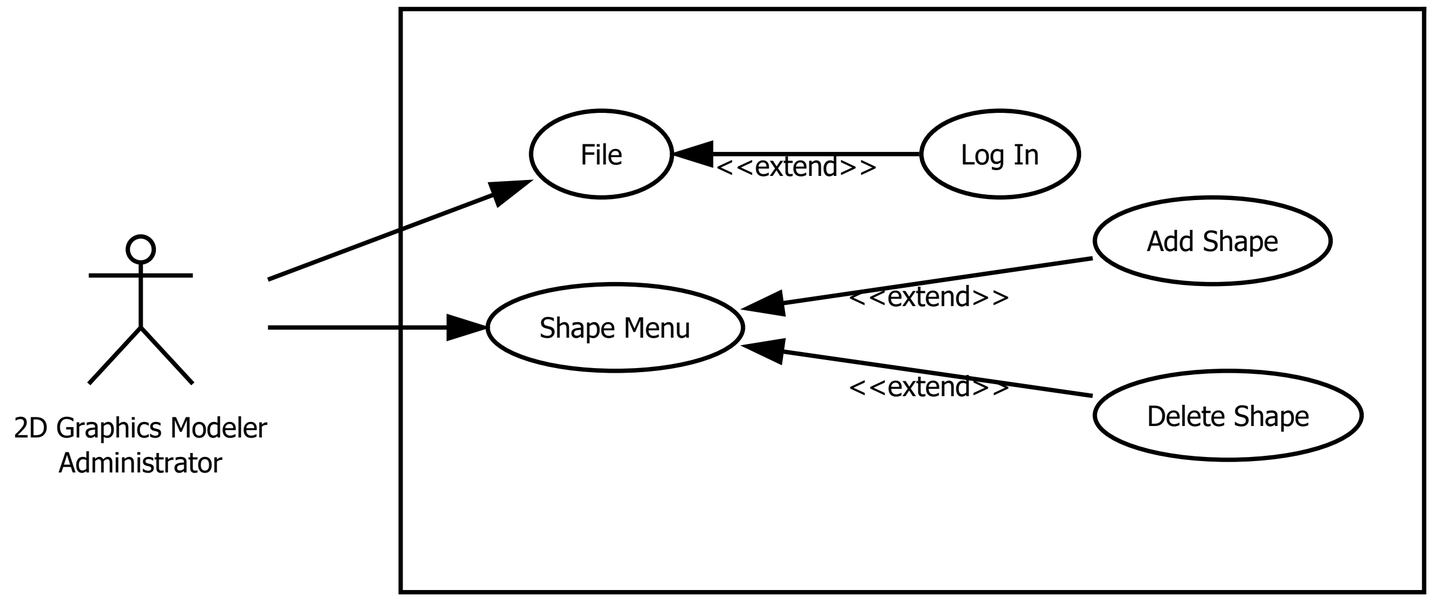
<<Component>>
Move Shapes
Use Case #01



UML Use Case 02 – Add or Remove Shapes

Use Case Number	02
Application	2D Graphics Modeler (Super Scrum)
Use Case Name	Add or Remove Shapes
Use Case Description	The actor can add or remove shapes including texts being rendered
Primary Actor	2D Graphics Modeler Administrator
Precondition	• Initialize shape list • Actor is logged in as Administrator
Trigger	The actor can select the option to add or remove shapes on the canvas
Basic Flow	1. Actor navigates to the File tab in the menu bar 2. Actor selects Log In in the drop down menu 3. Actor logs into Administrator account 4. Actor navigates to the shape menu 5. Actor adds a shape from the shape menu 6. Actor selects shape from canvas 7. Actor deletes shape from shape menu
Alternate Flows	• Actor can create a shape or use an existing one from the shape menu

<<Component>>
Add or Remove Shapes
Use Case #03



UML Use Case 03 – User Testimonial

Use Case Number	03
Application	2D Graphics Modeler (Super Scrum)
Use Case Name	User Testimonial
Use Case Description	The actor can contribute a User Testimonial
Primary Actor	2D Graphics Modeler user or customer
Precondition	• Customer account
Trigger	The actor can navigate to the comments section and create a testimonial
Basic Flow	1. Actor navigates to About tab in the menu bar 2. Actor selects Comments in the drop down menu 3. Actor selects the Click Me option to log in 4. Actor logs into their Customer account 5. Actor types in their opinion in the text box 6. Actor clicks Post 7. Testimonial is persistent between executions
Alternate Flows	• Actor can create a testimonial as a guest without needing to log into a customer account • Actor logged in with a Customer account can append extra feedback to their original comment

<<Component>>
User Testimonial
Use Case #03