**TekenElement.cs**

A completely new file, which exists to make each drawn element into an object with properties that need to be kept track of. This includes a list of points, the used Tool, Color, optional text and optional Hoek. These last two are only used in TekenElementen with Tool text, as other objects don’t have text and get their points updated on rotate instead of just being drawn **as if** they were rotated.

**TekenElementMaster.cs**

New file, which keeps track of all TekenElementen. It has a member variable of TekenTekenElementLijst, which keeps track of the TekenElementen currently displayed on the screen, and a member variable of WeggehaaldLijst, which keeps track of the removed elements and is used by the Redo() and Undo() methods.

VerwijderElement() method removes the first TekenElement that is detected by looping through TekenElementLijst in reverse. This is done to make sure the the TekenElement that is on top, which is the last one added in the list, gets removed first. The detection is done by method Geraakt().

ElementOmhoog() and ElementOmlaag() methods are called by using the hoog or laag tools. They check the TekenElementLijst in reverse, to make sure they select the highest TekenElement visible. If there is a TekenElement selected, they will move it up or down the list respectively.

ZoekDragElement checks whether there is a TekenElement clicked, then returns this TekenElement to the method which called it.

Method Roteer() rotates all TekenElementen. It transforms the bitmap’s coordinate system from (0, 0) in the topleft to (0, 0) in the middle. It then moves each quadrant to rotate them clockwise, as would be expected in a bitmap rotate. In addition, if the Tool used for this TekenElement is ‘tekst’, it updates the Hoek property depending on whether it has already made a full rotation or not.

The Geraakt() method returns a bool, which is true if the Point p in the argument overlaps TekenElement te in the argument. It uses a switch case to select the correct method, depending on which Tool is used in the provided TekenElement.

Method TekstGeraakt() measures the string of the TekenElement, if the provided Point p is in this measurement, it will return true. An absolute value is not necessary here, as a text will always be measured from the top right to the bottom left.

Method KaderGeraakt() checks if a non-filled in square is clicked. It checks whether the provided Point p is within the circumference of the rectangle by calculating the start and end point and getting the absolute value of the difference between provided Point p and the coordinates of the rectangle, then returns true if this difference is less than 5 pixels to allow for a slight misclick.

Method VlakGeraakt() is a less sophisticated version of KaderGeraakt(), as it does not need to check for the circumference. If Point p is anywhere in the square, it will have been clicked.

Method OvaalGeraakt() calculates the circumference of an Ellipse, then checks whether Point p is around this circumference. The distance allowed for misclick is still 5, the last step just removed a square root to make the algorithm ever so slightly more efficient.

Method DiscGeraakt() is, again, a less sophisticated version of OvaalGeraakt. There is no need to calculate the circumference, so the entire ellipse is counted as clicked. This means there is only need to calculate where the surface of the ellipse is located.

Method PenGeraakt() checks if any of the points is clicked within 5 pixels. This is done by using the formula for the distance between two points. The removal of the square root is done once more in this method.

**Program.cs**

Initiated SchetsEditor out of Application.Run() to make it accessible to other classes.

**SchetsEditor.cs**

Increased window size to accommodate new tools.

Added ‘open’ option to menu.DropDownItems and its corresponding function to open SchetsPlus XML files.

Added method Gewijzigd() to keep track of changes in the currently active MDI child.

**SchetsWin.cs**

Added public bool ‘gewijzigd’ to keep track of changes in each MDI child separately.

Made member variable ‘schetscontrol’ public, so it could be accessed by the ‘OpenXml’ method, which needs to set the object list to the newly created SchetsControl.

Added a confirmation window to check if the user wants to save before quitting, don’t save before quitting, or cancel quitting. This is done keeping track of a Boolean property named ‘Wijzig’ which is specific for each SchetsWin instance, so that they don’t interfere with each other. If ‘Wijzig’ is true and thus a change has been made to the image, the MessageBox shows. If the user presses ‘No’, meaning they don’t want to save before quitting, the window is closed. If the user presses ‘Cancel’, meaning they don’t want to quit at all, the MessageBox is hidden. If the user presses ‘Yes’, meaning they want to save before quitting, the Opslaan method is called to save the currently displayed image as either SchetsPlus XML or any image type.

Added ‘Opslaan’ method, which checks if the current file is already saved. If it is, it will call either SchrijfXml or SaveBitmap, depending on the extension of the fileNaam. If it is not yet saved, it will call the OpslaanAls method.

Added OpslaanAls method, which asks the user if the currently active MDI child should be saved as SchetsPlus XML or as an image. Calls SchrijfXml or SaveBitmap respectively.

Added SchrijfXml method, which writes the entirety of the TekenElementMaster TekenElementLijst to an XML file.

Added SaveBitmap method, which saves the currently displayed bitmap to the previously selected location. Also checks if the Close method should be called. Calls the OpslaanPopup() method.

Added OpslaanPopup method, which notifies the user whenever the file is saved as either XML or image.

Added Wijzig property, which links to member variable wijzig, to keep track of the changes in the currently selected MDI child.

Added tools ‘OvaalTool’, ‘VolOvaalTool’, ‘BovenopTool’, ‘OnderopTool’ and ‘MoveTool’ to ‘deTools’, so that they are displayed on the lefthand side of the screen and in the ToolStripMenu, making them selectable. Also added icons to the icon folder.

Enlarged the SchetsWin size to accommodate the newly added tools.

Added EventHandler FormClosing to the afsluiten method, so that the method gets called when the enclosing SchetsEditor is closed as well.

Added ‘opslaan’ and ‘opslaan als’ buttons to the file menu, linking to their corresponding methods.

Removed standard colors and added a ‘Kies kleur’ DropDownItem to open the ColorDialog.

Removed colors and ComboBox from ActieButtons, as they are not used anymore. Added buttons for save, undo, redo and color selection actions, linking to their corresponding methods.

**SchetsControl.cs**

Set default value of member variable penkleur to Color.Black, as there would otherwise be no default color selected using the new color system.

Added a property which gets the TekenElementMaster corresponding to each instance of SchetsControl.

Added Undo() and Redo() methods, which remove the lastly added TekenElement and replace it again, respectively.

Added OpnieuwTekenen() method to replace Invalidate(), as this would not update the new TekenElementMaster TekenElementLijst.

Removed old color selection methods and replaced them with VeranderKleur(), which deals with all color selection with a ColorDialog.

**Schets.cs**

Added member variables newWidth and newHeight, to keep track of whether the width and height of the Schets window changed. This is used by TekenElementMaster.Roteer(), to calculate the new position of each TekenElement respectively.

Added member variable TekenElementMaster tem, to keep track of the TekenElementMaster corresponding to each Schets individually.

Added Ophalen property, which returns tem. This allows SchetsControl to access Schets’ TekenElementMaster.

Added method TemSchrijven(), which allows tem to be overwritten after OpenXml has made a new TekenElementMaster TekenElementLijst.

Added newWidth and newHeight to VeranderAfmeting() method, to update them whenever the Schets window is resized.

Added a second Teken() method, which takes a list of TekenElement as an argument, to (re-)draw each TekenElement in TekenElementMaster’s TekenElementLijst. Also flips text respective to its Hoek property.

**Tools.cs**

Added a member TekenElement and TekenElementMaster to StartpuntTool, which allows it to set the properties of each TekenElement and add it to TekenElementMaster’s TekenElementLijst.

Set most of the properties of TekenElement in MuisVast, as most of the information is already known at that point.

Set the Punten property of TekenElement in MuisLos, as the second point is known at that point.

Set Letters property of TekenElement in TekstTool’s Letter() method, adding each letter to the string when they’re typed.

Rewrote TweepuntTool’s MuisLos() method completely, by adding the second point to TekenElement’s Punten property, having two Teken() methods called instead of a Bezig() and a Compleet() method. One Teken() method will draw the TekenElement as its not placed yet, the other will draw it once MuisLos() is called.

Added OvaalTool() and VolOvaalTool() by drawing an outlined or filled in Ellipse respectively in a rectangle bounding box.

Rewrote PenTool to continuously add points to TekenElement’s Punten, then drawing a line between each point in the meantime.

GumTool is now a subclass of ISchetsTool instead of PenTool. It’s also completely rewritten to get the current TekenElementMaster and calling its VerwijderElement() method, then passing the updated TEM to Schets’ updated Teken() method, which will redraw all elements in TEM’s TekenElementLijst.

Added BovenopTool and OnderopTool, which call TEM’s ElementOmhoog() and ElementOmlaag() methods respectively. This makes it so the clicked TekenElement is moved all the way up or all the way down by moving it to the end or start of the list.

Added MoveTool, which allows the user to drag and move each TekenElement separately. When MuisVast() is called, it saves the current mouse location and gets the TEM and possibly selected TekenElement. It checks whether a TekenElement was clicked by calling TEM’s ZoekDragElement, which from the end to the start through the TekenElementLijst and returns the first TekenElement it registers as clicked. The actual dragging of the element is done by calculating the offset of the mouse current location to where the mouse first was detected, saved in Point hier. It then moves all Punten saved in TekenElement by the offset, redraws the entire Schets and updates the mouse. The ever so slight delay between the calculation and updating of the mouse position means there will be a slight difference between Point hier and the current mea.Location. This means the selected TekenElement will move smoothly along with the mouse.