Toelichting

Voor een volledig overzicht van alle directe veranderingen in de code, zie het commit log: https://github.com/mrexodia/SchetsPlus/commits/master. Dit bestand is enkel een overzicht met enkele ontwerpkeuzes toegelicht.

Ellips

Voor het tekenen van een ellips zijn twee nieuwe klassen toegevoegd. De ene voor de werking van de tool en de andere als representatie van een ellips (zie de volgende sectie voor meer informatie).

Het nieuwe gummen

Om het nieuwe gummen te realiseren is ervoor gekozen om naast de `ISchetsTool`-hiërarchie een hiërarchie te maken met als basistype `SchetsObject`. De tools maken volgens instanties van hun objecten aan in een lijst die de huidige schets representeert (`LijnTool` voegt bijvoorbeeld een `LijnObject` toe aan de lijst). Zie `SchetsPlus.pdf` voor een overzicht van de nieuwe klassenhiërarchie.

`SchetsObject` heeft een virtuele methode `Geklikt()`, waarin een object kan bepalen of erop geklikt is. De algemene `Geklikt()`-methode tekent elk object los in een lege, transparante bitmap en kijkt vervolgens of de pixel op (x,y) niet meer transparant is. Waar mogelijk wordt deze methode vervangen door een meer wiskundige manier om te bepalen of er op een object geklikt is. Voor objecten met een lijndikte (rechthoeken, ellipsen en lijnen) is ervoor gekozen om een marge van 2 pixels te nemen. Deze marge kan zo nodig worden aangepast door een constante aan te passen. Voor tekst is er geen marge en voor gevulde objecten ook niet.

`LijnObject` gebruikt een formule van http://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line#Line_defined_by_two_points om de afstand van een punt tot een lijn te berekenen.

`EllipsObject` gebruikt een formule van http://www.analyzemath.com/calculus/Integrals/area_ellipse.html om een waarde te berekenen die wat zegt over waar in de ellips geklikt is.

De `GumTool` loopt simpelweg (in omgekeerde volgorde) over de lijst van objecten en roept vervolgens de `Geklikt()`-methode aan om te bepalen of er op een object geklikt is. Als dit het geval is wordt het object uit de lijst verwijderd.

Bij `PenObject` is ervoor gekozen om de hele lijn in één keer te verwijderen. `PenObject` is niets anders dan een lijst van `LijnObject`en. Bij `TekstObject` is ervoor gekozen om de hele tekst in één keer te verwijderen.

Elk `SchetsObject` heeft een methode genaamd `Teken()`, die het object in een `Graphics`-structuur tekent.

Opslaan en teruglezen (.schets bestand)

Voor het opslaan van de lijst van objecten is gekozen om gebruik te maken van serialization. De `SchetsObject` klassen zijn serializeerbaar gemaakt en de methodes in `ObjectSerializer` worden gebruikt om de lijst te lezen/schrijven van/naar een XML-bestand. Dit XML-bestand wordt vervolgens met GZip gecomprimeerd. Op deze manier hoeven er geen eigen bestandsformaten ontwikkeld worden, wat tijd (en een hoop fouten) scheelt. Bij het openen van een foutief bestand zal er een foutmelding getoond worden.

Om te detecteren of er veranderingen zijn in de lijst van objecten wordt gebruik gemaakt van de methode `GetHashCode()`. De hash van de volledige lijst wordt vergeleken met de hash na de laatste keer opslaan/openen. Als deze ongelijk zijn zal er aan de gebruiker worden gevraagd of hij wil opslaan.

Exporteren

Voor het exporteren naar gangbare afbeeldingsformaten (BMP, JPG, PNG) wordt gebruik gemaakt van de `Bitmap.Save()`-methode. Exporteren wordt gedaan met een apart menuitem.

Undo/Redo knop in het menu

Om ervoor te zorgen dat de gebruiker alle acties ongedaan kan maken, maar ook weer opnieuw kan doen wordt er gebruik gemaakt van een zogenaamde `UndoList`. Deze lijst heeft (min of meer) dezelfde functionaliteit van de `List`-klasse, maar houdt daarnaast ook alle veranderingen bij. Voor een uitgebreidere toelichting, zie deze post (de klasse is wel ontwikkeld door ons): http://mrexodia.cf/coding/2014/11/01/UndoList

Andere toevoegingen/wijzigingen

Met de `LayerTool` kunnen objecten met de linkermuisknop helemaal bovenop de stapel worden geplaatst en met de rechtermuisknop helemaal onderop de stapel.

De `PickerTool` kan gebruikt worden om een kleur te selecteren door deze aan te klikken in de schets.

Het is mogelijk om de kleur 'Anders...' te selecteren die een `ColorDialog` opent, waar de gebruiker een groot aantal kleuren kan kiezen en ook zelf kan samenstellen.

Er is een control voor de dikte van de pen toegevoegd. Deze dikte gaat van 1-20, maar het is mogelijk om het maximum aan te passen met een constante.

Om het lettertype van de `TekstTool` aan te passen is een knop toegevoegd, die een `FontDialog` opent waar je het lettertype kan selecteren.

De `TekstTool` heeft nu de mogelijkheid om letters te verwijderen met backspace.

Het is mogelijk om het programma in zijn geheel te vertalen. Hierbij is gebruik gemaakt van resources in meerdere talen. Omdat we geen vertaalbureau tot onze beschikking hebben zijn alleen de talen Nederlands en Engels beschikbaar. Het kost echter weinig moeite om vertalingen toe te voegen. De taal wordt automatisch aangepast aan de taal van het OS. De standaard is Engels. Door de vertalingsmogelijkheid was het nodig om een methode `Icoon()` aan de ISchetsTool interface toe te voegen omdat de `ToString()` methode nu taalafhangkelijk is.

Andere (kleine) zaken

- De oorsprongkelijke broncode is geformatteerd met behulp van Visual Studio.
- De icoontjes zijn bewerkt zodat het er allemaal (in onze mening) wat leuker uitziet. Verder heeft de applicatie een icoontje. Alle icoontjes zijn met de hand gemaakt.
- roteren van de lijst door middel van een virtuele methode `Roteer` in `SchetsObject`.
- De titel van het MDI child window wordt de bestandsnaam als je opslaat.