Wijzigingen

1: CirkelTool en VolCirkelTool toegevoegd aan Tools.cs, grotendeels gebaseerd op RechthoekTool en VolRechthoekTool. Verder ook icoontjes toegevoegd in de map en namen bij deTools lijst gezet.

2: Om de file te exporteren/op te slaan hebben we een nieuwe methode Opslaan() gemaakt in klasse SchetsControl waarmee je in verschillende bestandstypes, die we in een lijst hebben gestopt, het plaatje kan opslaan. Dit kan als bmp, gif, jpeg, jpg, png, en eventueel zouden we gemakkelijk nog extra bestandstypes toe kunnen voegen door ze in de lijst te zetten. Hiervoor bevindt zich een dropdownitem genaamd ‘Opslaan als’ onder de menu kop ‘file’ zodra een schets-window is aangemaakt, waar je uit de verschillende filetypes kan kiezen. De methode Opslaan() pakt eerst de form, dan de titel van die form, en slaat het bestand op in een lokale folder ‘drawings’, of ‘drawingtxt’, als we het bestand als textfile opslaan. De titel van de form is ook waar we overigens de bestandsnaam ook in opslaan, als we een textfile openen met de Openen() methode. Deze bestandnaam kan de gebruiker ook veranderen door middel van een inputbox die tevoorschijn komt als de gebruiker op ‘Rename’ drukt onder het kopje ‘File’, dit doen we ook met een try-catch voor foutafhandeling bij bijvoorbeeld karakters die niet mogen worden ingevoerd in de bestandsnaam.
Voor de waarschuwing die de gebruiker krijgt als die het schets-window wil sluiten, met niet-opgeslagen wijzigingen, hebben we de twee methodes die beide ‘afsluiten()’ heten aangepast. Hiervoor moesten we eerst de functie laten kijken of de schets veranderd is ten opzichte van het laatste moment wanneer die is opgeslagen, hiervoor gebruikten we in de Schets-klasse een kopie van de bitmap die wordt geüpdatet wanneer het bestand wordt opgeslagen, dit door hulp van een vergelijking. Om de waarde van deze boolean-vergelijking te kunnen checken in de afsluiten() methodes, moesten we per methode wat anders doen.
In de klasse SchetsWin, hebben we eerst een methode IsGewijzigd gemaakt die in de klasse Schets naar de variabele zoekt die bijhoudt of de bitmap veranderd is, en die methode IsGewijzigd roepen we in een if-statement op in de methode afsluiten, die een waarschuwingsbericht in een messagebox stuurt naar de gebruiker, als de bitmap ook veranderd blijkt te zijn. De klasse SchetsEditor is een parent van de klasse SchetsWin, aangezien we met de methode nieuw() een SchetsWin() object maken, dus hebben we ervoor gekozen om eerst te checken of er überhaupt een child schets-window is aangemaakt, en dan van die child de methode IsGewijzigd oproepen om te kijken of de bitmap is veranderd, en dan weer hetzelfde waarschuwingsbericht stuurt als in de klasse SchetsWin.

3: Deze criteria vereiste vrij veel, we hebben de taak als volgt opgesplitst:
3.1: Eerst was het nodig om een nieuwe klasse "Doodle" aan te maken om de gegevens van getekende objecten in op te slaan, en ook een lijst “doodles” waarin de Doodle objecten werden opgeslagen. De lijst doodles staat in SchetsControl.cs zodat elk canvas zijn eigen lijst heeft die toegankelijk is in andere bestanden met ‘s.doodles’.
3.2: Het aanmaken van Doodle objecten en vervolgens toevoegen aan doodles gebeurt allemaal in Tools.cs onder de bijbehorende Tools.
Voor alle TweepuntTool tekeningen konden we op dezelfde manier een Doodle aanmaken

en toevoegen aan doodles, dit werkt vrij vanzelfsprekend.
Voor PenTool tekeningen hebben we ervoor gekozen om de hele lijn op te slaan als een Doodle object met een lijst Punten<Point> die alle segmenten bijhoudt. Op deze manier kan je makkelijk de hele lijn uitgummen. Hiervoor moesten we wel de MuisDrag functie herschrijven zodat hij al direct de lijn segmenten tekent, anders zou hij namelijk constant MuisLos aanroepen en dat maakte het moeilijk om één eindpunt te krijgen voor de hele lijn.
Voor TekstTool hebben we ervoor gekozen om voor elke individuele letter een Doodle object te maken, dit geeft meer vrijheid bij het uitgummen.
3.3: Nadat Doodle en doodles goed werkten konden we beginnen met het herschrijven van GumTool. Eerst was het nodig om de functie 'bool Raak(Doodle d, Point p)' te maken.
Raak() checkt eerst of de Doodle het PenTool type heeft:
Zo wel, dan gaat het alle punten in de Puntenlijst van de Doodle langs en maakt het een rechthoek om elk segment met een kleine marge en checkt het voor elk rechthoekje of het gegeven punt p er binnen valt.
Is het geen PenTool, dan maakt het een rechthoek om de hele Doodle en checkt het of het punt er binnen zit.
3.4 De nieuwe GumTool kan nu Raak() gebruiken. Elke keer dat er met het GumTool geklikt wordt gaat het de hele lijst doodles af en checkt het met Raak() of de klik op een van de Doodle objecten zit, het begint hierbij bij het einde van de lijst en stopt met zoeken zodra er eentje raak is. Op deze manier wordt alleen het 'bovenste' object weggehaald.
Als Raak() true teruggeeft, wordt het Doodle object verwijderd uit de lijst en het veld wordt opnieuw gemaakt met de nieuwe lijst.

4: Voor het opslaan als Doodle tekst, hebben we in klasse SchetsControl een nieuwe methode OpslaanAlsDoodleText() gemaakt op basis van de methode Opslaan(), in principe werkt die hetzelfde, maar moet de doodle als tekst worden geschreven in een textfile, in plaats van dat de bitmap als plaatje wordt opgeslagen. Voor elke doodle maken we hierbij een nieuwe lijn aan, zodat deze file ook weer makkelijk te openen is en in de doodle van de schets terug te zetten. Een probleem was dat we een lijst punten moesten toevoegen, om te zorgen dat de PenTool doodle goed werd opgeslagen met meer tussenpunten, en niet alleen 'start' en 'finish'. Daarom hebben we een lijst 'puntenlijst' die we als string aan de lijst 'doodleLines' aan het einde van de Doodle-eigenschappen toevoegen. Deze hele lijst 'doodleLines' voegen we toe aan een nieuwe file die we voor nu lokaal in de Solutionfolder 'drawingtxt' stoppen. Hieruit halen we ook weer de textfiles als de gebruiker verder wilt met een tekening, dit gebeurt in Openen(). Deze functie slaat ook weer de naam van het bestand dat geopend is op in de titel van de SchetsWindow, zodat die ook later weer goed opgeslagen kan worden word, ook als de gebruiker de naam van de file aanpast tijdens het verder veranderen van de tekening. Met Openen roepen we ook een functie TekenDoodle() die de Doodles weer tekent, en in een lijst stopt, waarmee de gebruiker weer verder kan. Deze functie hadden we eerst als één lange functie, maar hebben we opgeplitst, zodat we met het gummen, en eventueel als we nog een Undo() methode toe zouden voegen, dat we per 1 lijn ook een doodle makkelijker aan een zelfde of andere lijst toe kunnen voegen of juist eruit halen, en ook voor 'het nieuwe gummen' bleek dit handig te zijn.

Extra: We hebben naast de onderdelen 1t/m4 ook nog een aantal extra's ingebouwd. We hebben ten eerste bij veel methodes een Debug.WriteLine() toegevoegd, om makkelijker te kunnen vinden waar iets eventueel fout gaat (zonder een Console window hoeven aan te maken), of bijvoorbeeld een file succesvol is opgeslagen, of überhaupt het programma

initieel wel opstart. We hebben ook een control toegevoegd waarmee de gebruiker een andere lijndikte kan instellen, die de dikte van de pen, en de randen van de rechthoeken en ellipsen verandert. Dit doormiddel van een nieuwe dropdown menu onder aan het schets-window, we hebben hiervoor gekozen, zodat de gebruiker gemakkelijk van lijndikte kan veranderen, zonder door de menukopjes te hoeven navigeren. Anders zou de gebruiker te gemakkelijk per ongeluk op een ongewenste waarde kunnen klikken.