DN4

Arbeitsblatt: DNET1

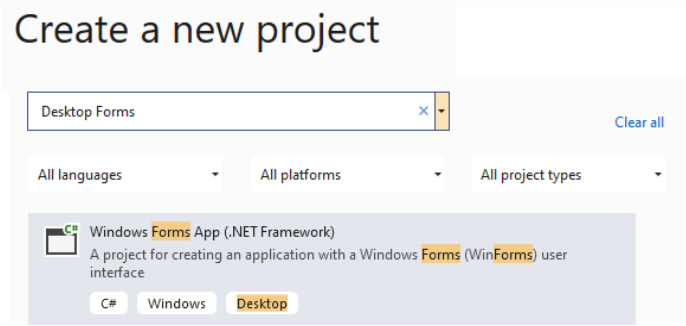
Name: _____ **Kurznamen:** _____

Verloren im Weltall, Planetensimulation

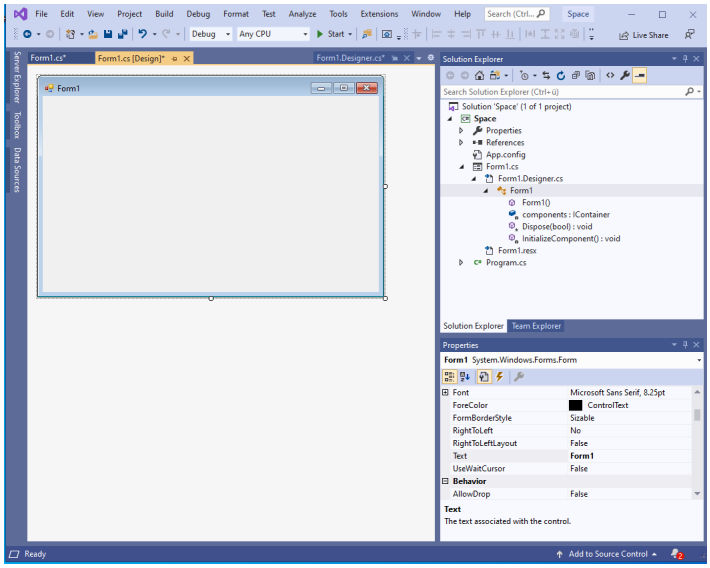
Die Bewegungen der Planeten können u.a. durch die sogenannten Kepplerschen Gesetze bestimmt werden. Diese gehen aber von einem Zentralgestirn aus, in dem der Grossteil der Masse konzentriert ist, was für unser Sonnensystem ja auch der Fall ist. Gäbe es dieses Zentralgestirn nicht, kämen kompliziertere Formeln zur Anwendung. Ab drei Massen existiert keine geschlossene Formel mehr, sondern die Bewegungen lassen sich nur noch numerisch bestimmen. Es sollen hier die Plantenbahnen ohne die Sonne berechnet werden. Wir ignorieren den Umstand, dass ohne diese es vermutlich niemanden geben würde, den diese Berechnung interessiert.

Aufgabe 1

Zuerst wollen einen attraktiven Hintergrund für unsere Simulation vorbereiten. Erstellen Sie dafür ein Projekt: *Windows Forms (.NET Framework)* oder einfach *Windows Forms*.

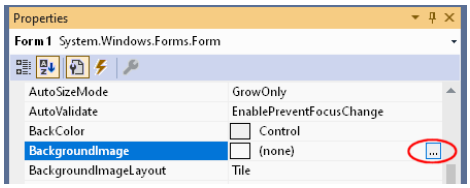


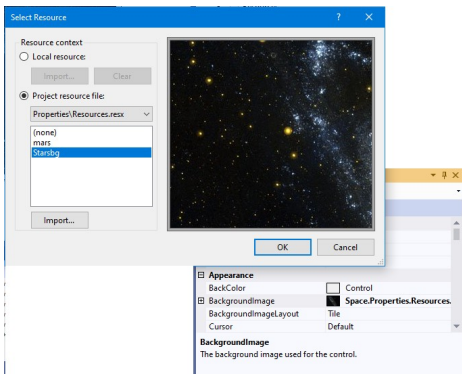
Und fügen Sie die Klassen Orb, Planet und Spaceship dem Projekt hinzu.



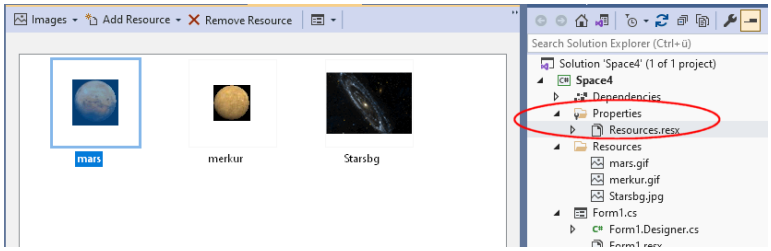
Bilder als Ressourcen

Setzen Sie den Sternenhintergrund über die Properties des Forms (als Resource). Fügen Sie bei dieser Gelegenheit gleich noch die Bilder der Planeten als weitere Ressourcen hinzu (über den gleichen Dialog)





Es wird automatisch eine *typisierte Ressource* angelegt, deren Inhalt sie via Namen ansprechen können (siehe unten). Sie können auch später neue Ressourcen via Drag-and-Drop hinzufügen. Das .resx File beschreibt den Inhalt der Ressource.



Zeichnen eines Planeten mittels Knopf

Erstellen Sie für Testzwecke einen Knopf in Ihrem Form, der den Mars an der Stelle 100 100 in halber Grösse zeichnet. Einfach ein Knopf via Drag-and-Drop aus der Toolbox und Doppelklick um eine Event Verarbeitungsmethode zu generieren.

Sie können dann nach dem entsprechenden Import auf die Bilder zugreifen (Projektname ist der Name den Sie für Ihr Projekt gewählt haben):

```
using <ProjectName>.Properties;
```

In der Event Verarbeitungsmethode soll dann das Bild gezeichnet werden:

```
Image image = Resources.mars;
```

oder via Namen zugegriffen werden

```
Image image = (Bitmap)Resource.ResourceManager.GetObject("mars");

Graphics g = this.CreateGraphics();
g.DrawImage(image, 100, 100, image.Width/2, image.Height / 2);
```

Automatisches (Neu-)Zeichnen aller Planeten

Leiten Sie von Himmelskörper *`Orb`*, die neuen Klassen *`Planet`* und *`Spaceship`* ab (separate Klassen weil wir Spaceship später noch anpassen werden). In der Basisklasse Orb wird die Bewegung der Himmelskörper implementiert, welche unten implementiert wird.

Statt über einen Knopf sollen beim *Neuzeichnen des Forms* alle Himmelskörper (über ihre *Draw* Methode) automatisch gezeichnet werden. Laden Sie dafür das Bild des Himmelskörper (via Namen) in der Draw Methode und zeichnen Sie ihn zunächst an vordefinierten Positionen.

Hinweise:

- Erstellen Sie zwei Klassen `Planet` und `Spaceship`, die beide von Orb erben und die (abstrakte) Draw Methode überschreiben.
- Führen Sie eine Liste der Himmelskörper (Orb) in Ihrem Form ein.
- Überschreiben Sie die OnPaint Methode im Form.
- Ähnlich wie bei Java existiert für Forms eine Methode `OnPaint`, die automatisch aufgerufen wird und die überschrieben werden kann:

  ```
  protected override void OnPaint(PaintEventArgs e) {
            Graphics g = e.Graphics;
  ```

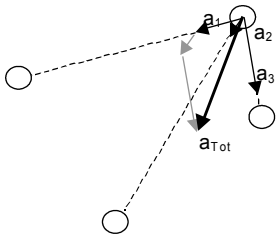
- Die Orb Liste wird bei der Erstellung des Form durch die 4 Himmelskörper initialisiert und in der OnPaint Methode wird die Draw Methode aller enthaltenen Objekte aufgerufen
- Es wird über die Klasse Graphics gezeichnet welches beim Aufruf der OnPaint Methode als Feld des PaintEventArgs mitgegeben wird: e.Graphics.

Aufgabe 2

Die Bewegungen sollen nun berechnet und angezeigt werden. Die Anziehungskraft zwischen zwei Massen berechnet sich aus (wie Ihnen sicherlich aus der Physik noch geläufig ist) $F = G * (M_1 * M_2) / r^2$. Die für eine Beschleunigung einer Masse notwendige Kraft berechnet sich aus $F = a * M$. Daraus lassen sich die resultierenden Beschleunigungen der einzelnen Massen in unserem System berechnen. Die resultierende Beschleunigung lässt sich einfach durch vektorielle Addition der einzelnen Beschleunigungskomponenten berechnen. Für die Berechnung der Beschleunigungskomponente hat sich folgende Formel bewährt:

$$\vec{a}_1 = |a_1| * \frac{\vec{r}_1}{|\vec{r}_1|}$$

Einheitsvektor in Richtung r



Gehen Sie im Beispiel von einem relativen Massenverhältnis der Planeten von 100 (Jupiter) zu 5 (Mars) zu 4 (Merkur) aus. Die Enterprise habe eine für ein Raumschiff enorme relative Masse von 1 (wegen des Warp Antriebes und der geladenen Antimaterie).

Hinweise:

- Kopieren Sie die Vector Klasse in Ihr Projekt (mit Namespace DN3)
- Die Gesamtbeschleunigung einer Masse setzt sich aus den Einzelbeschleunigungen zusammen. Achten Sie aber dabei, dass sie zuerst *sämtliche* Beschleunigungen und neuen Geschwindigkeiten berechnen und erst anschliessend die Massenpunkte verschieben (zwei Durchläufe notwendig).
- Für die Animation kann die Timer Klasse verwendet werden, das einfach mittels drag-and-drop aus der Toolbox in das Form gezogen werden kann.
- Setzen Sie im Konstruktor der Form Klasse die entsprechende Timer Properties

```
this.timer1.Enabled = true;
this.timer1.Interval = 50;
this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
```

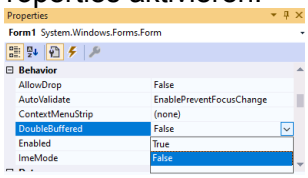
- Fügen Sie eine private void timer1_Tick(object sender, EventArgs e) Methode hinzu, die einen `Refresh` Aufruf des Neuzeichnen auslöst und damit einen nächsten Schritt der Planetensimulation.
- Folgende Anfangswerte sollten den Mars um den Jupiter kreisen lassen.

```
space.Add(new Planet("jupiter", 600, 400, -0.038, 0, 100));
space.Add(new Planet("mars", 600, 600, 3.8, 0, 1));
```

- Folgende Anfangswerte sollten ein chaotisches Verhalten zeigen

```
space.Add(new Spaceship(600, 230, 3, 0, 1));
space.Add(new Planet("jupiter", 600, 400, -0.099, 0, 100));
space.Add(new Planet("mars", 300, 400, 0, 1.5, 4));
space.Add(new Planet("merkur", 200, 200, 0, -1.5, 4));
```

- Um Flickern zu vermeiden, können Sie das sog. Double Buffering über die Forms Properties aktivieren:



Abgabe:

Praktikum: DN4.1

Datei: Orb.cs