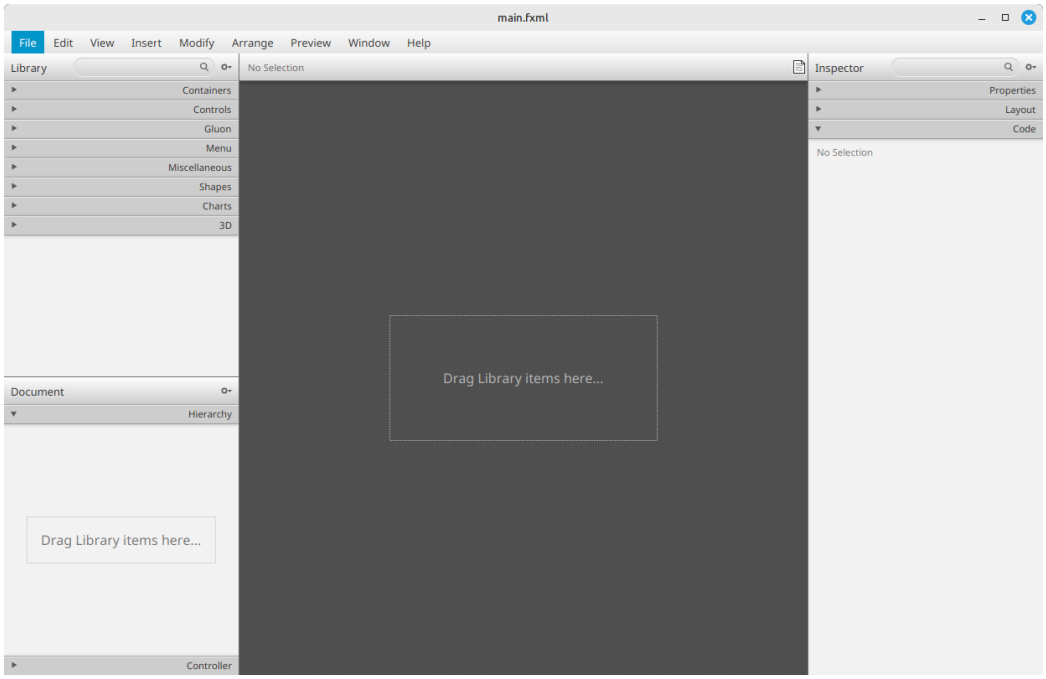
JavaFX mit SceneBuilder extern

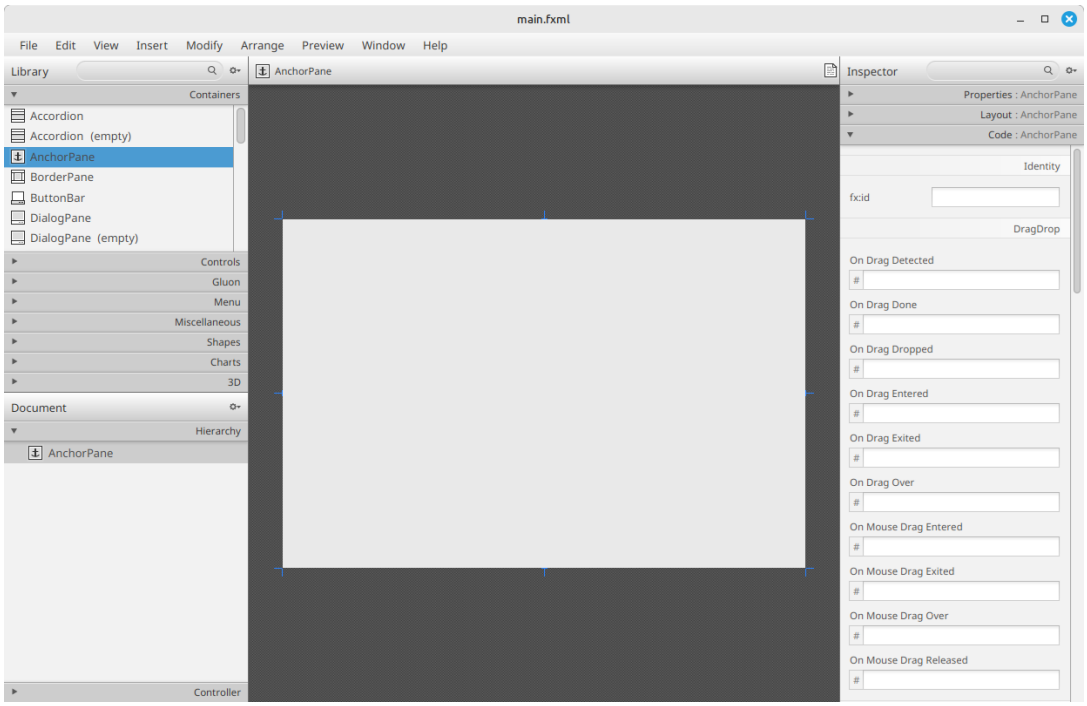
new File



FXML ist, wie der Name schon sagt, eine XML-Sprache, d.h. mit einer Baumstruktur von Elementen und ihren Attributen. Im Designer sehen wir links das Menu für mögliche Elemente. Wenn wir ein Element eingefügt haben, haben wir rechts das Menü für mögliche Attribute zu dem Element. Die FXML-Datei sollten wir im Resource-Verzeichnis unseres Projektes speichern, ebenso die png-Dateien für unsere Karten.

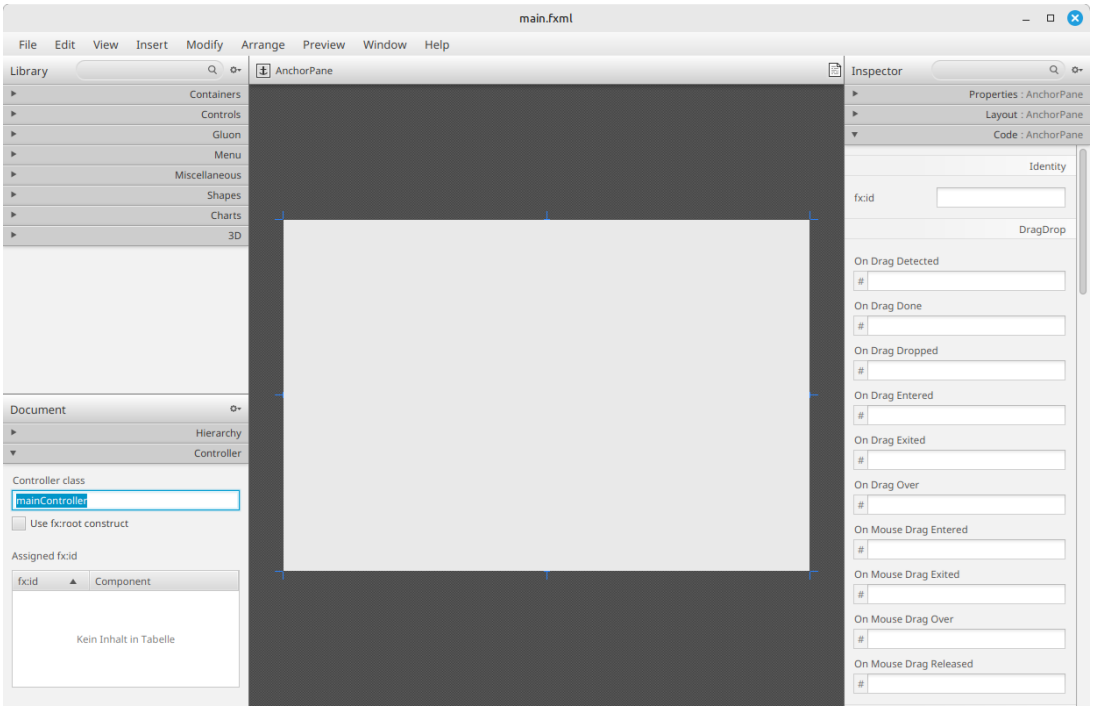
Basislayout wählen

Im Menu Container wählen wir zunächst unser Basis-Layout aus, in diesem Fall ein Anchor Pane. In diesem Layout werden die Controls an dem Punkt verankert, an dem wir sie im Designer anlegen.



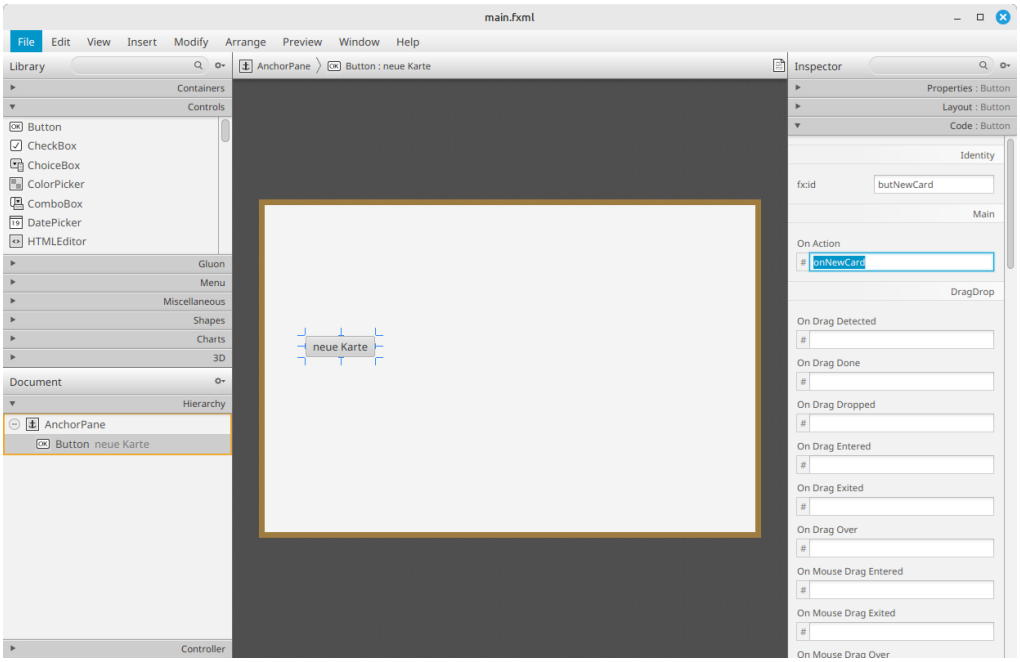
Zuordnen der Controller Klasse

Bevor wir nun Controls in unserem Layout anordnen, sollten wir im Controller Menu (links ganz unten) die gewünschte Controller Klasse angeben. In dieser Klasse werden alle benannten Controls als Variablen angelegt und alle Callbacks als Funktionen mit den entsprechenden Events als Übergabeparameter.



Einfügen eines Controls (hier Button)

Wir fügen einen neuen Button ein und geben ihm im rechten Menü-Bereich über Properties einen neuen Text. Interessanter ist es im rechten Menü-Bereich Code. Wir geben dem Button eine FX-Id und eine Callbackfunktion onAction ...

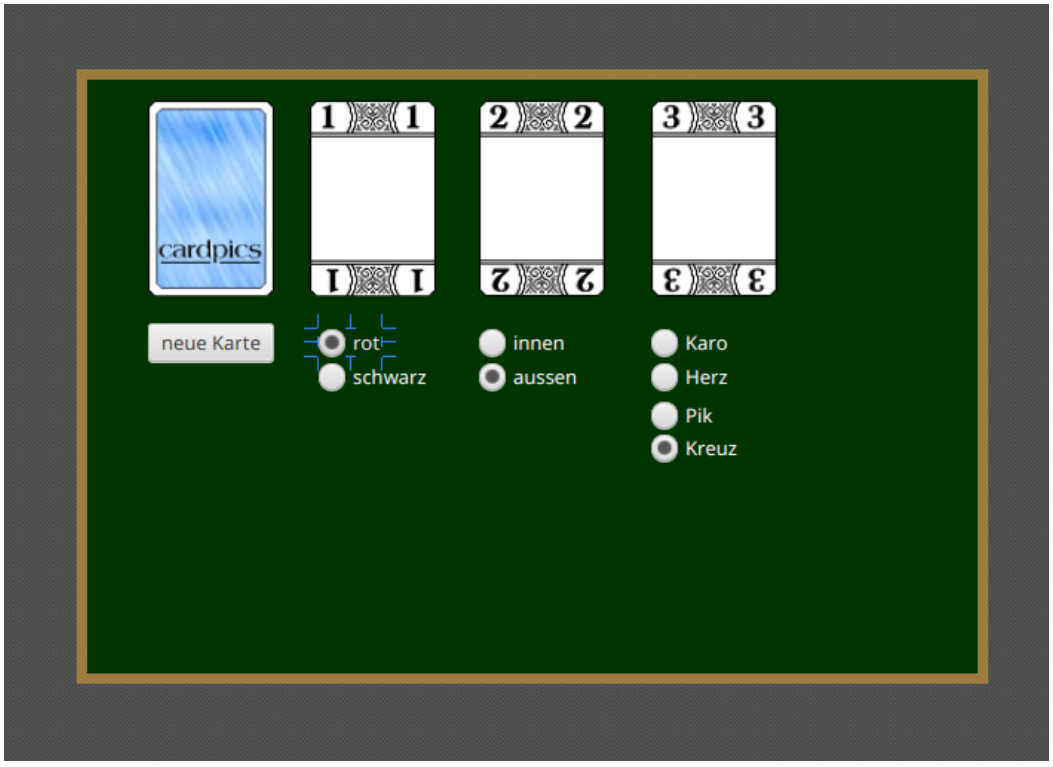


...und stellen fest, dass im Beispielcode des Controllers (View/Skeleton) eine Membervariable für den Button generiert wurde und eine Callback-Function für das onAction-Event.

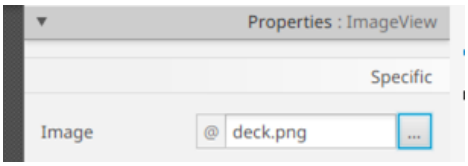
Sample Skeleton for 'main.fxml' Controller Class

```
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;

public class mainController {

   @FXML
   private Button butNewCard;

   @FXML
   void onNewCard(ActionEvent event) {

   }

}
```

So können wir im Designer zunächst die Oberfläche unseres Spiels entwerfen:

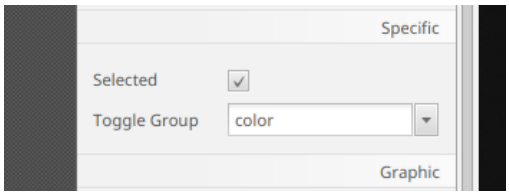


Die Controls zur Anzeige der Karten sind ImageViews. Die darzustellende png-Datei kann bei den Properties angegeben werden



Damit der Inhalt dieser ImageViews vom Programm geändert werden kann, muss im Code-Bereich (rechtes Menu) eine FXId vergeben werden.

Radiobuttons werden rechts im Properties-Menü einer ToggleGroup zugeordnet. Das sorgt dafür, dass immer nur ein Radiobutton ausgewählt sein kann.
Der Aufgabe entsprechend sind die Radiobuttons unterschiedlichen ToggleGroups zugeordnet.



Der automatisch generierte Controller-Code für dieses Design würde folgendermaßen aussehen:

Sample Skeleton for 'main.fxml' Controller Class

```
package cardgame;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.ToggleGroup;
import javafx.scene.image.ImageView;

public class mainController {

   @FXML
   private ToggleGroup Farbe;

   @FXML
   private Button butNewCard;

   @FXML
   private ToggleGroup color;

   @FXML
   private ImageView ivPos2;

   @FXML
   private ImageView ivPos3;

   @FXML
   private ImageView ivStapel;

   @FXML
   private ImageView ivpos1;

   @FXML
   private ToggleGroup pos;

   @FXML
   void onNewCard(ActionEvent event) {

   }

}
```

Copy Save as... Java Comments Full

Diese Basis Controller-Class können wir in unserem Projektverzeichnis speichern.

Jetzt müssen wir unsere start-Methode dahin gehend ändern, dass wir unsere Controls nicht mehr selber bauen, sondern die FXML-Datei und den Controller verwenden.

```
@Override//
public void start(Stage stage) throws IOException {

    //  Laden der FXML-Datei
    FXMLLoader fxmlLoader = new FXMLLoader
                                (mainApp.class.getResource("main.fxml"));

    //  Verbinden der FXML-Datei mit der Controller-Class
    mainController ctrl = (mainController) fxmlLoader.getController();

    //  Laden der FXML-Datei in die Szene
    Scene scene = new Scene(fxmlLoader.load());

    stage.setScene(scene);
    stage.setTitle("Cardgame");
    stage.show();
}
```

In mainController:

```
public Image cards[] =new Image[52];        // Karten
public void initCards() {
    for (int i = 0; i < 52; i++) {
        String c = "card" + i + ".png";
        String pfad = getClass().getResource(c).toString();
        cards[i] = new Image(pfad);
    }
}

void onNewCard(ActionEvent event) {
    Random rand = new Random();
    int t = rand.nextInt(52);
    ivDeck.setImage(cards[t]);
}

void checkQ1(int t){
    RadioButton selRadioButton=(RadioButton)color.getSelectedToggle();
    String textRadiobutton = selRadioButton.getText();
    if (t > 26 ^ textRadiobutton.equals("rot")){
       // Frage1 richtig beantwortet!
       // Karte auf pos1
       ivPos1.setImage(getAktCard());
       // Karte merken
       cardPos1 = t;
       // nächste Frage
    }
    else resetGame();
}
```