**Tagesablauf (Tage ab Sonntag):**

1. Sonntag: Einführung mit wxWidgets. Projekt mit Buttons und einfachen Objekten. Anfangen des Sudoku-Projekts und Programmierung des Bretts.
2. Montag: Brett-Klick Funktion hinzugefügt. Erste Version des Backtracking Algorithmus geschrieben. (funktioniert nicht)
3. Dienstag: Verzweifelt an der Backtracking Funktion für die Erstellung des Spielfeldes.
4. Mittwoch: Solver und einfacher creator geschaft.
5. Donnerstag: Feinschliff & Timer; wenn länger als 0.5sek; dann nochmal Spielfeld erschaffen

**To-Do:**

* Volles Sudoku Spielfeld generieren
* Bestimmte Zahlen des Spielfeldes entfernen und immer wieder einen Lösungsalgorithmus ausführen.
* Lösungsalgorithmus schreiben
* Lösungsalgorithmus kann als sichtbar oder unsichtbar eingestellt werden
* Placeselection weg machen und die platzierten nur bei einem empty feld berücksichtigen.
* Checken, ob das Feld vollständig ist, bei User Eingabe
* Die Felder hervorheben, die die gleiche Nummer haben (vlt. Farbe ändern)
* Die Lösungsdings auslagern und auch im Generationscheck die Lösungsfunktion benutzen
* Vlt. die Generierungsfunktion überarbeiten, da immer noch fehlerhaft

**Mögliche Wege zum Ziel:**

Generierung:  
V1: Wie im Video, an die Stellen, an denen am wenigsten mögliche Zahlen platziert werden können, die nächste platzieren. Davor checken ob diese Möglichkeit überhaupt funktioniert!

V2: Fast wie Algorithmus bis jetzt, nur das zuerst alle 1, 2, … gesetzt werden. Diese werden zufällig nach den Regeln gesetzt. Wenn es sich nicht ausgeht, wird so lange zurückverfolgt, bis es geht.

Lösungsalgorithmus:  
Wie jetzt, nur das von 1-9 hochgezählt wird, wenn der Wert 9 erreicht und nicht platziert werden kann, wird zurückverfolgt (bool auf false, wenn die nächste Pos belegt wird, bool auf true). Hierbei muss irgendwie Buch geführt werden, welche Zahlen schon probiert wurden.