Fußballspiel

Inhalt des Spiels

Wer schießt in diesem spannenden Zwei-Spieler-Duell mehr Tore?

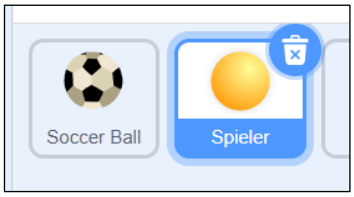
Voraussetzungen

Diese Anleitung benutzt viele *Nachrichten* und *eigene Blöcke*. Bevor du dieser Anleitung folgst, solltest du verstehen, wie Blöcke und Nachrichten funktionieren.

Außerdem solltest du dich generell mit Scratch auskennen, also zum Beispiel wissen, wie man Figuren und Kostüme erstellt oder Variablen anlegt

Bühne & Figuren

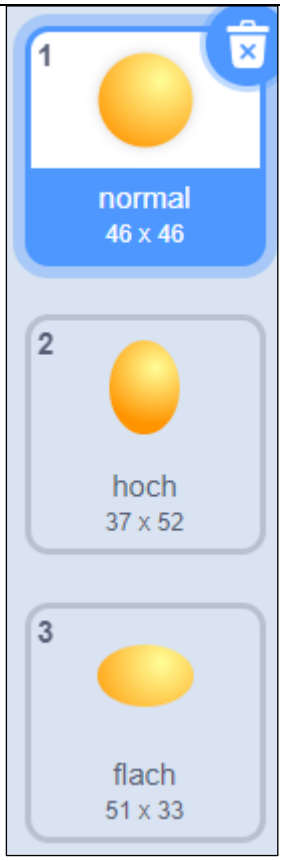
Setze zunächst das Bühnenbild auf "*Soccer 2*". Diese Bühne zeigt ein Spielfeld von der Seite und hat sogar schon zwei Tore.



Als Spielfigur wähle zunächst den "*Soccer Ball*" und setze dessen Größe auf 50. Dann eine Figur "*Ball*" für unseren Spieler. Benenne diese auf "*Spieler*" um und reduziere die Größe auf 60.

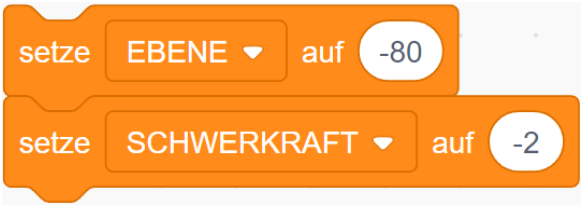
Lege neue *Kostüme* für den *Spieler* an. Lösche dazu alle Kostüme bis auf das erste, welches du zweimal duplizierst.

Das erste Kostüm nennen wir "*normal*" und lassen es unverändert.
Das zweite Kostüm nennen wir "*hoch*" und ändern die Form so, dass sie etwas schmäler und höher ist.
Das dritte Kostüm nennen wir "*flach*" und ändern die Form so, dass sie etwas breiter und flacher ist.



Vorbereitungen

In der Figur *Soccer Ball* legen wir nun einen neuen Block namens "*Game Loop*" an. Dieser enthält eine fortlaufende Schleife, welche immer wieder die Nachricht "*Bewegung*" an alle sendet. Klicke einmal auf den Block, um ihn laufen zu lassen - er ist dann gelb umrandet.

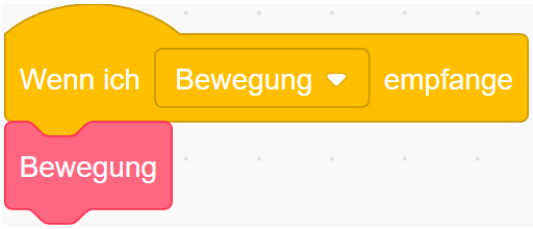


Lege außerdem eine neue Variable für alle Figuren an, welche du "*EBENE*" nennst, und auf **-80** setzt. -80 ist diejenige y-Position, an der die Spielfiguren stehen sollen - mitten am Fußballfeld. Wir möchten später nicht ständig "-80" eingeben, deshalb nutzen wir eine sogenannte "*Konstante*" - genau das gleiche wie eine Variable, bloß, dass du den Wert während des Spielverlaufs nicht änderst. Um den Überblick zu behalten schreiben wir Konstante immer groß.

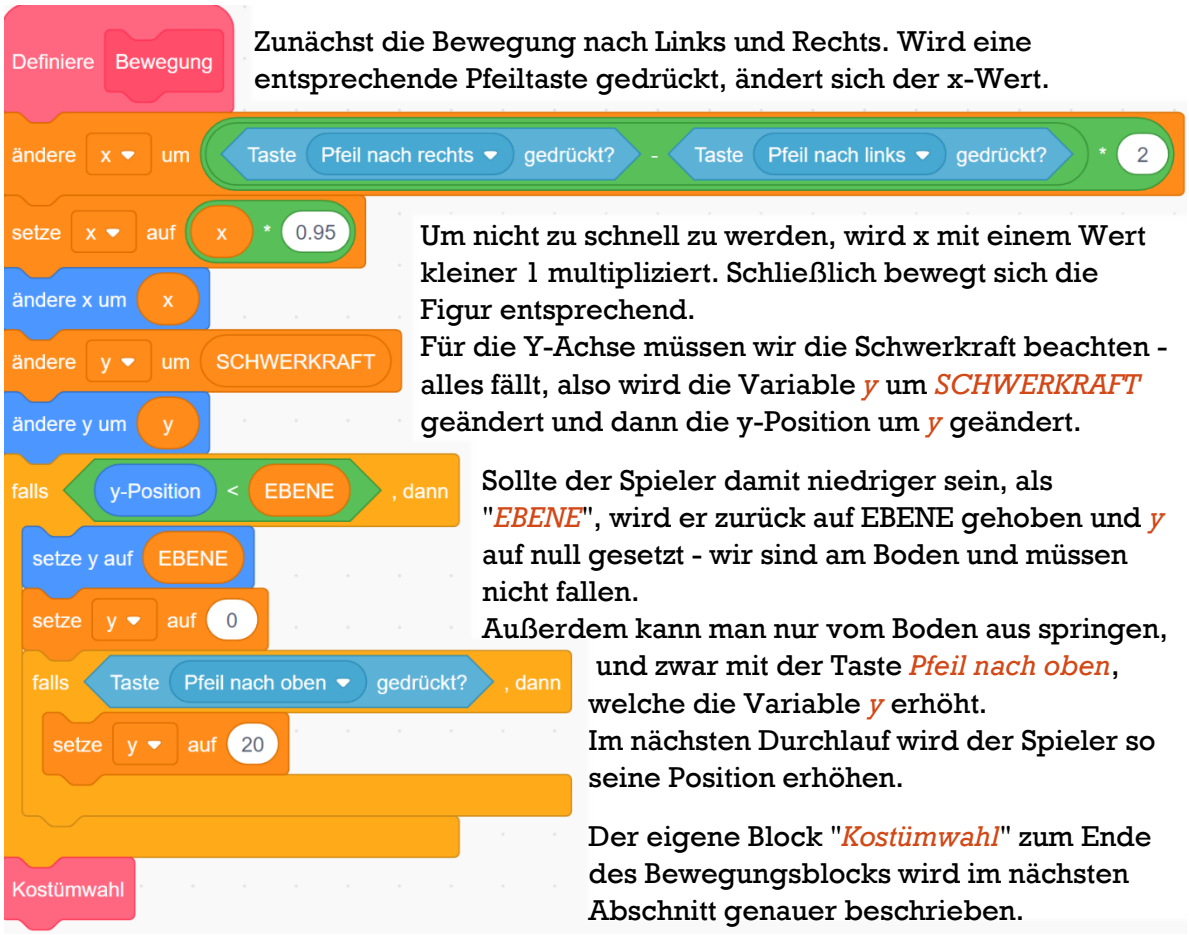
Eine weitere Konstante "*SCHWERKRAFT*" soll den Wert **-2** bekommen. Um einen Wert einzustellen, musst du entweder einen Rechtsklick auf die Anzeige der Variable im Spielbereich machen, sie auf einen Schieberegler ändern und diesen ändern, oder den entsprechenden Block "setze auf" entsprechend konfigurieren und anklicken.

Spielerbewegung

In der Figur *Spieler* definieren wir einen neuen Block "*Bewegung*", welcher *ohne Bildschirmaktualisierung* ausgeführt werden soll. Dieser Block soll immer abgespielt werden, wenn die Nachricht "*Bewegung*" empfangen wird.



Wir benötigen außerdem zwei Variablen, "*x*" und "*y*", welche - wichtig - *nur für diese Figur* genutzt werden sollen.

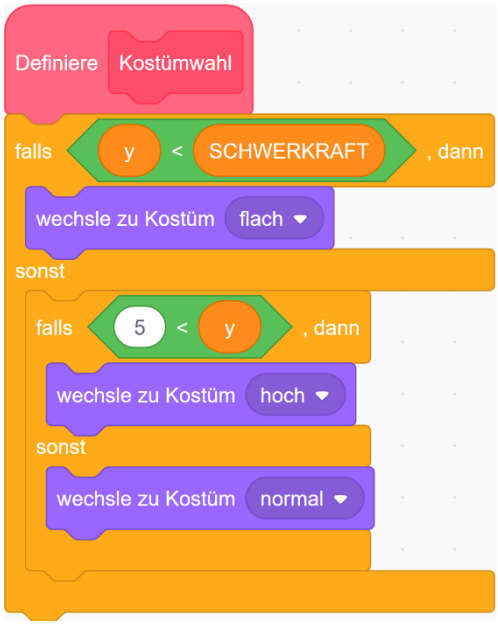


Zunächst die Bewegung nach Links und Rechts. Wird eine entsprechende Pfeiltaste gedrückt, ändert sich der x-Wert.

Um nicht zu schnell zu werden, wird x mit einem Wert kleiner 1 multipliziert. Schließlich bewegt sich die Figur entsprechend.
Für die Y-Achse müssen wir die Schwerkraft beachten - alles fällt, also wird die Variable *y* um *SCHWERKRAFT* geändert und dann die y-Position um *y* geändert.

Sollte der Spieler damit niedriger sein, als "*EBENE*", wird er zurück auf EBENE gehoben und *y* auf null gesetzt - wir sind am Boden und müssen nicht fallen.
Außerdem kann man nur vom Boden aus springen, und zwar mit der Taste *Pfeil nach oben*, welche die Variable *y* erhöht.
Im nächsten Durchlauf wird der Spieler so seine Position erhöhen.

Der eigene Block "*Kostümwahl*" zum Ende des Bewegungsblocks wird im nächsten Abschnitt genauer beschrieben.



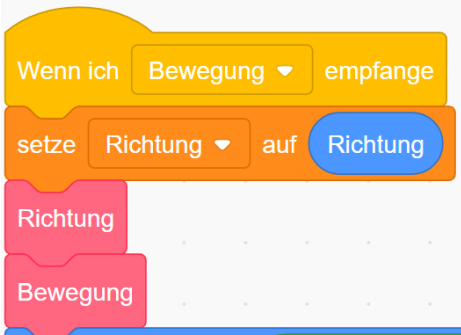
Nun kannst du einen eigenen Block "*Kostümwahl*" erstellen. In diesem wird die *y*-Variable abgefragt.
Ist sie geringer als die *Schwerkraft*, fällt die Figur und soll *flach* erscheinen.

Ist y Positiv, so springt die Figur gerade, und bekommt das Kostüm *hoch*. Aber anstatt mit 0 zu vergleichen, nutzen wir lieber *größer 5,*

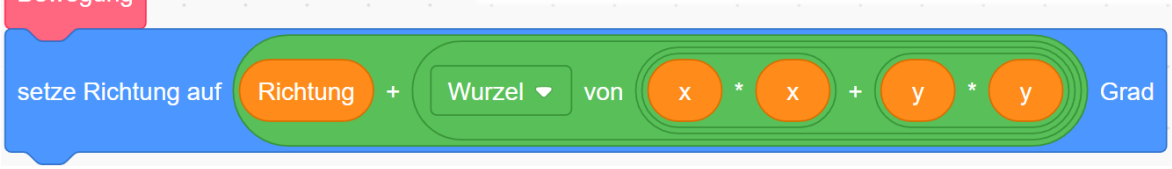
so bleibt die Figur *normal* nahe null, also wenn sie am Boden ist oder ganz oben im Sprung (auch Apex genannt).

Probiers mal aus - deine Figur sollte sich steuern lassen, springen und dabei ein ganz kleines bisschen die Form verändern.

Spielball



Auch der Spielball benötigt eigene Variablen für *x* und *y*, sowie für die *Richtung*.
Den Bewegungscode wollen wir hier auf zwei Teile aufteilen - "*Richtung*" und "*Bewegung*".
Vor diesen Codestücken möchten wir die aktuelle Richtung des Balls in der Variable "*Richtung*" zwischenspeichern, und danach die Richtung des Balls setzen.

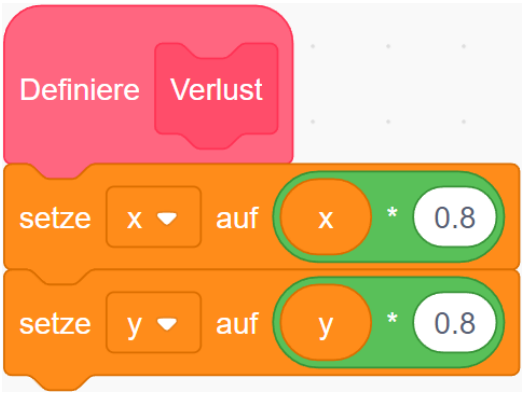


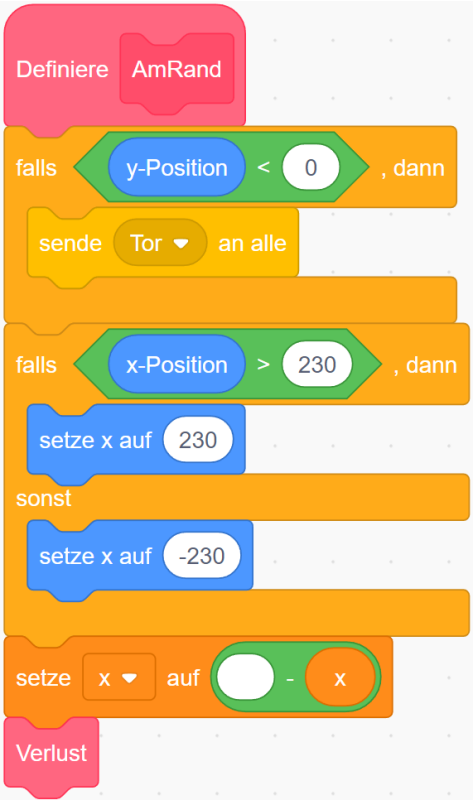
Du kannst einfach zurück auf "*Richtung*" setzen, dann dreht sich der Ball nie. Setzt du auf "*Richtung + 10*" dann wird er sich konstant gleichmäßig drehen. Die angegebene Formel kennst du vielleicht als "Pythagoras" - sie sorgt dafür, dass der Ball so schnell rotiert, wie er sich gerade bewegt.

Die Bewegung soll darauf achten, wo der Ball gerade ist. Wir ändern zunächst die *x-Position* um die *x-Variable* und prüfen, ob der *Betrag* der Position so größer 230 ist. "Betrag" entfernt das Minus von negativen Zahlen - ein Betrag größer 230 bedeutet also, die X-Position ist entweder größer als 230 oder kleiner als -230 - also "*Am Rand*", ein Umstand, den wir in einem weiteren Block behandeln wollen.

Bei der *y-Position* müssen wir ebenso wie beim Spieler darauf achten, ob *EBENE* unterschritten wird. Allerdings soll der Ball nicht bloß landen, sondern gleich wieder hochhüpfen, deshalb kehren wir die y-Variable um und nutzen den eigenen Block "*Verlust*".

Um welchen Verlust geht es im Block "Verlust"? Reibungswiderstand. Wenn du einen Fußball schießt, wird er irgendwo mal stehen bleiben. Wenn du ihn durch die Luft schießt kommt er aber weiter, als wenn du ihn am Boden rollen lässt - der Boden macht den Ball langsamer.
Deshalb wird hier sowohl die *x-Variable* als auch die *y-Variable* um ein Fünftel reduziert.





Im Block *AmRand* wissen wir, der Ball ist ganz links oder ganz rechts.

Bei einer y-Position unter 0 ist er im *Tor*, also senden wir eine entsprechende *Nachricht* aus und kümmern uns später woanders darum.

Wie auch am Boden soll der Ball nie über die Ränder bei 230 und -230 hinausschießen. Deshalb wird ein zu hoher Wert auf 230 zurückgesetzt, aber wenn der Wert zu niedrig ist auf -230 erhöht.

Auch die *x-Variable* wird umkehrt, um beim abprallen die Richtung zu wechseln. Weil auch hier der Ball Kontakt zu einer Fläche hat wird seine Geschwindigkeit wieder mit dem eigenen Block *Verlust* reduziert.

Damit kommen wir zum Block "*Richtung*". Dieser behandelt vor allem den Kontakt zum Spieler. Wird der *Spieler* berührt, so dreht sich der Ball zu ihm hin, dann um *180 Grad* - also von ihm weg.

Sinus und *Cosinus* sind Winkelfunktionen. Sie erlauben uns, die *Richtung* in die beiden Achsen *x* und *y* aufzuspalten.
Die neu anzulegende Konstante "*KICK*" gibt an, wie stark sich der Ball beschleunigen soll, wenn ein Spieler ihn trifft.

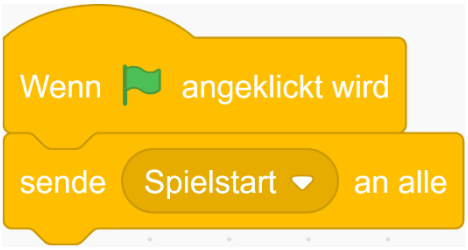
Außerdem versucht jeder Spieler, den Ball nach oben zu schießen, deshalb erhöhen wir die *Y-Variable* nochmal extra um 10.

Schließlich wollen wir die Schwerkraft nicht vergessen. Die Reduktion des *x-Wertes* ist quasi der Luftwiderstand - Wir haben ja im Block "*Verlust*" den Rollverlust durch Bodenkontakt abgebildet, aber auch in der Luft wird der Ball immer langsamer.

Nun hast du eine Spielfigur und einen Spielball.
Probiere mal aus, ob du den Ball kicken kannst.

Spiellogik mit Nachrichten

Bisher müssen wir den Block "*GameLoop*" durch Anklicken selbst aktivieren. Das möchten wir nun ändern. In der Figur "*Soccer Ball*" legen wir das kürzeste Skript der Welt an:
Wenn Fahne berührt sende Spielstart an alle.

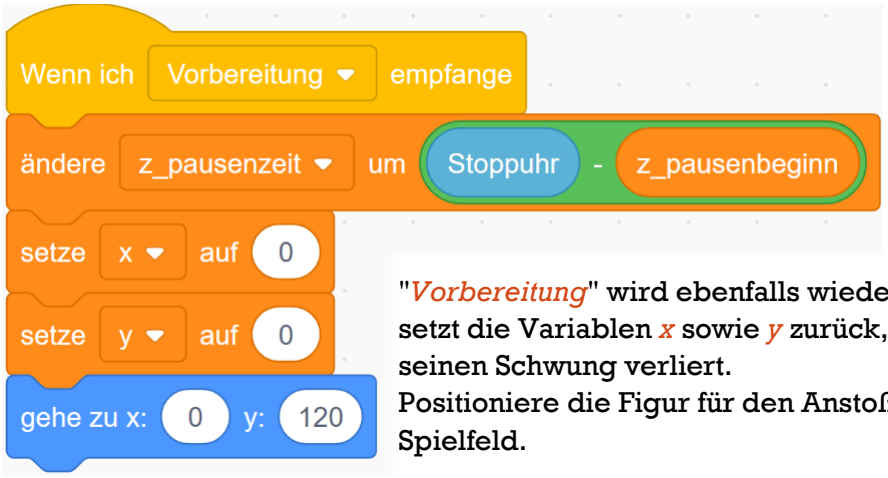


Ebenfalls im *Soccer Ball* kannst du die Nachricht "*Spielstart*" empfangen.

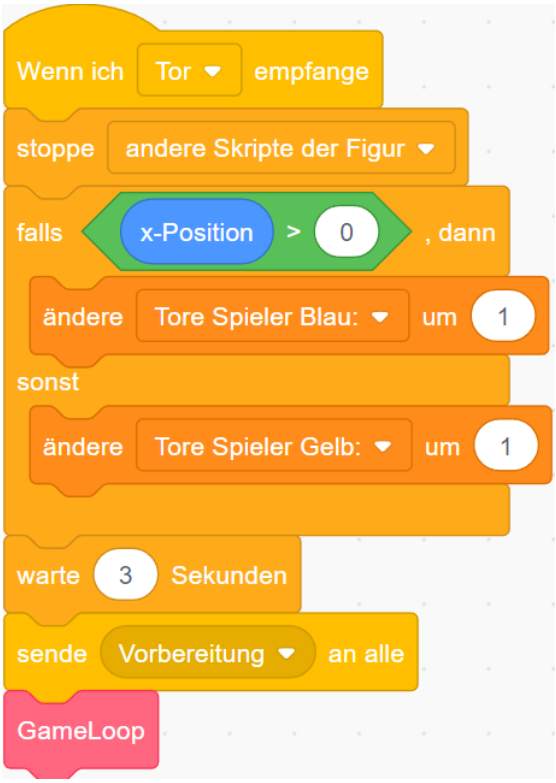
Zum Start des Spiels kannst du - *musst aber nicht* - die *Konstanten* einmal definieren. So kannst du dir sicher sein, dass sie nicht versehentlich umgestellt wurden.

Ansonsten wollen wir neue Torzähler als Variablen anlegen und hier für den Spielstart nullen. Diese beiden Variablen, „Tore Spieler blau:“ und „Tore Spieler Gelb:“ sollen auch während des Spiels am Boden der Spielfläche angezeigt werden.

Anschließend senden wir "*Vorbereitung*" an alle und beginnen die *GameLoop*.



"*Vorbereitung*" wird ebenfalls wieder abgefangen und setzt die Variablen *x* sowie *y* zurück, damit der Ball seinen Schwung verliert.
Positioniere die Figur für den Anstoß mittig über dem Spielfeld.

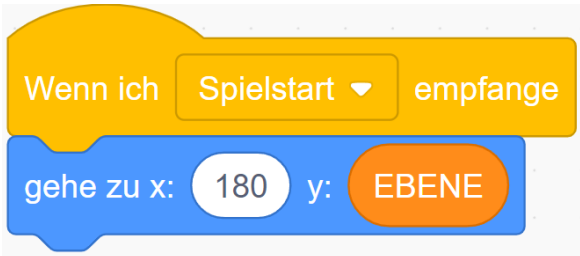


Die ältere Nachricht "*Tor*" wird ebenfalls als Trigger verwendet. Bei einem Tor soll zunächst alles andere aufhören - der Ball soll gar nicht erst weiterrollen.

Das Tor soll gezählt werden.
Dazu muss man wissen, auf welcher Seite der Ball ist, und die entsprechende Variable um eins erhöhen.

Nach einigen Sekunden Pause, in denen man Zeit hat, sein Tor zu feiern, wird das Spielfeld mit "*Vorbereitung*" zurückgesetzt und die *GameLoop* wieder aktiviert.

In Figur Spieler:



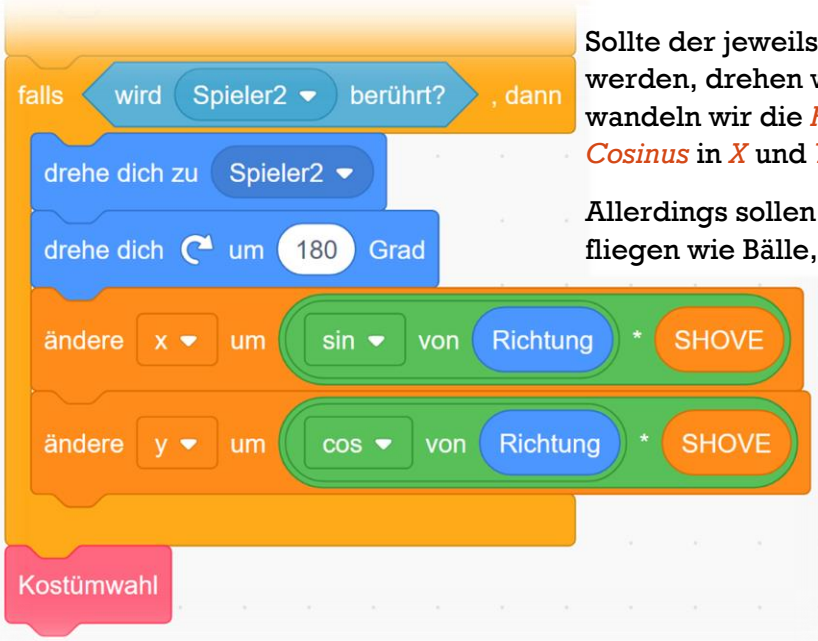
Auch "*Spieler*" kann diese Nachrichten empfangen und entsprechend an eine Startposition gehen, sowie die Variablen *x* und *y* nullen.

Zweiten Spieler anlegen

Da unsere Skripte für den Spieler komplett sind, können wir die gesamte Figur mit einem Rechtsklick duplizieren und so *Spieler2* anlegen.

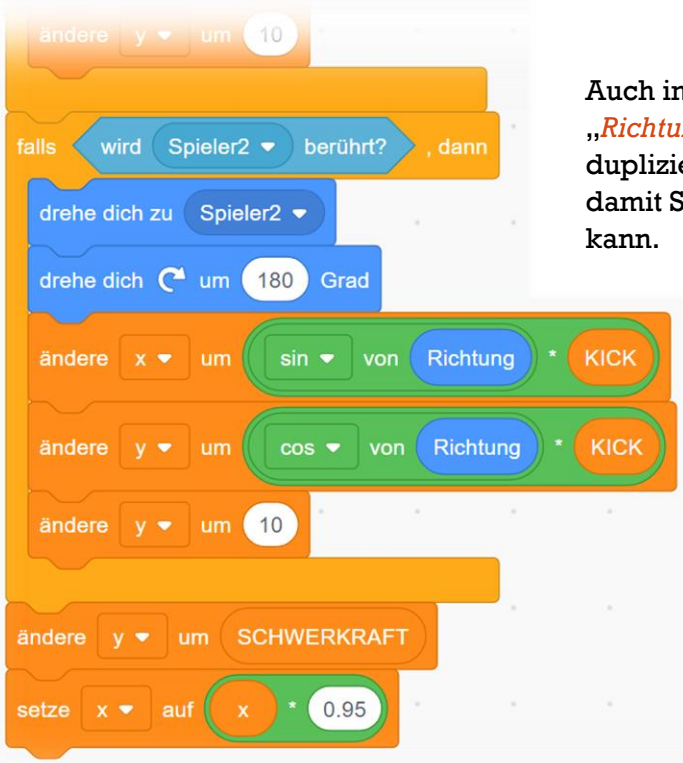
Geändert werden muss lediglich, dass Spieler 2 auf der anderen Spielfeldseite beginnt, und andere Tasten für die Bewegung nutzt - *A-W-D* bieten sich an. Natürlich sollte auch das Kostüm des zweiten Spielers umgefärbt werden.

Außerdem können wir uns nun überlegen, ob wir Körperkontakt erlauben. Nicht ganz realistisch, aber lustiger - Sowohl in Spieler als auch in Spieler2 ergänzen wir folgenden Code im Block "*Bewegung*":



Sollte der jeweils andere Spieler berührt werden, drehen wir uns von ihm weg. Dann wandeln wir die *Richtung* mit *Sinus* und *Cosinus* in *X* und *Y* um.

Allerdings sollen Spieler nicht so weit fliegen wie Bälle, daher wird statt mit "*KICK*" mit einer neuen Konstanten "*SHOVE*" multipliziert. *SHOVE* sollte auf 5 gesetzt werden.

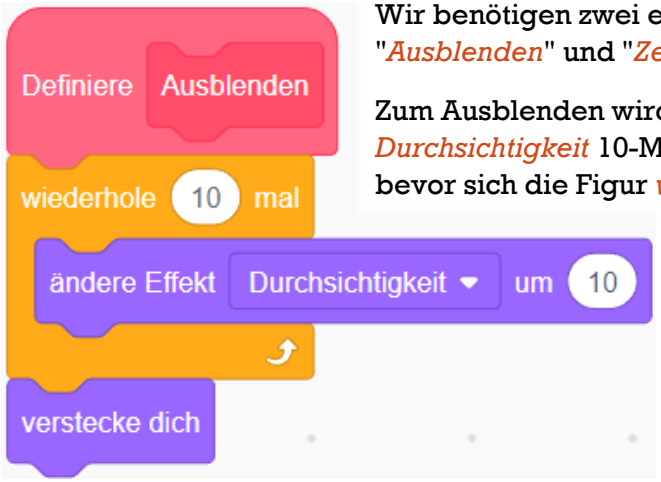


Auch im *SoccerBall* müssen wir im Block „*Richtung*“ das Codestück zum Kicken duplizieren und an Spieler2 anpassen, damit Spieler2 auch den Ball wegschießen kann.

Texte anzeigen mit Karten

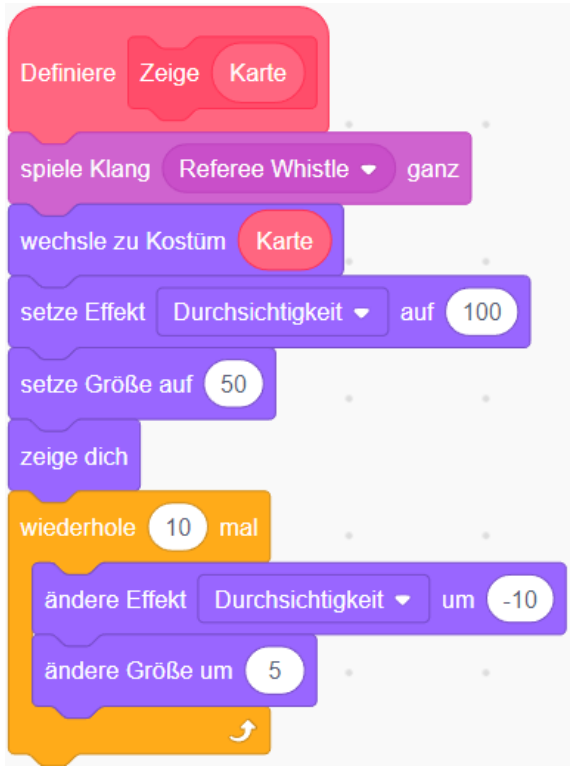
Damit die Spieler leichter mitbekommen, was passiert, legen wir eine neue Figur "*Karten*" an, welche wir selber zeichnen. Wir benötigen die Kostüme "*Tor*", "*Bereit*", "*Los*", "*Zeit*", "*Gelb siegt*", "*Blau siegt*" und "*unentschieden*", welche jeweils auch diesen Text enthalten.
Schreibe dazu mit dem *Textwerkzeug* (T) und nutze die Schrift „*Marker*“

Ergänze zudem den Klang "*Referee Whistle*".



Wir benötigen zwei eigene Blöcke, "*Ausblenden*" und "*Zeige (Karte)*".

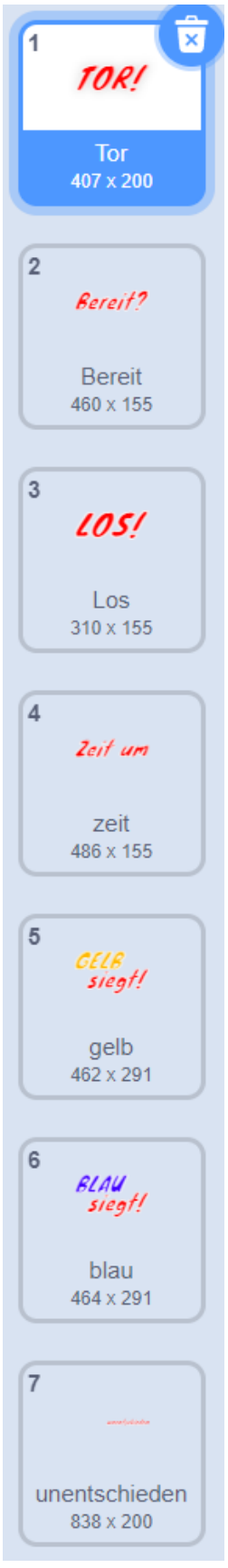
Zum Ausblenden wird die *Durchsichtigkeit* 10-Mal um 10 erhöht, bevor sich die Figur *versteckt*.

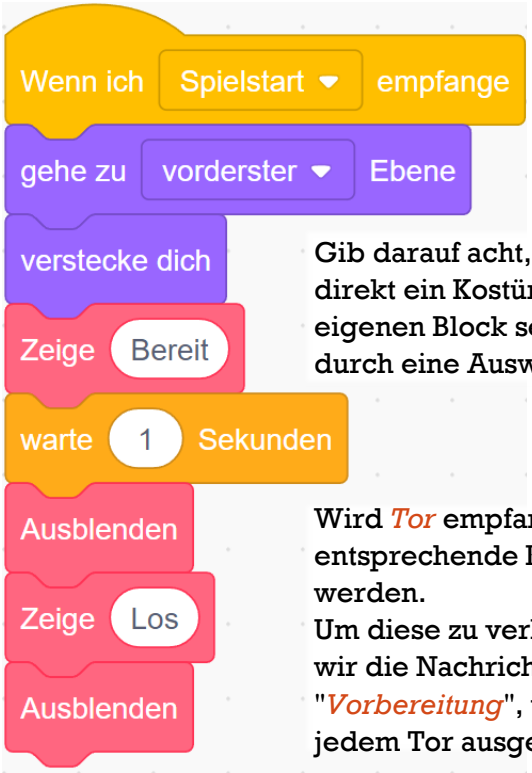


Wird eine Karte gezeigt, so erklingt das Schiripfeiffen und das Kostüm wird auf die entsprechende *Karte* gewechselt.

Setze die *Durchsichtigkeit* auf 100 und die Größe auf 50, bevor du diese Figur zeigst.

Dann lass Durchsichtigkeit und Größe im Verlauf von zehn Wiederholungen auf den Normalwert zurückkehren, um eine angenehme Einblendung zu erhalten.





Die Karten sollen zu*vorderst*, aber *versteckt* sein, deshalb machen wir das schon zum *Spielstart*.
Dann zeigen wir die Kostüme "*Bereit*" und, eine Sekunde später, "*Los*".

Gib darauf acht, keine Tippfehler zu machen, Wir wählen nicht direkt ein Kostüm, sondern senden nur einen Wert an einen eigenen Block senden, deshalb kann uns Scratch hier nicht durch eine Auswahl unterstützen.

Wird *Tor* empfangen, so soll die entsprechende Karte angezeigt werden.
Um diese zu verbergen, nutzen wir die Nachricht "*Vorbereitung*", welche nach jedem Tor ausgesandt wird.

Wir haben zwar noch kein Spielende einprogrammiert, aber wir können schon mal definieren, was passiert, wenn eine entsprechende Nachricht empfangen wird.
Zunächst wird das Kostüm "*Zeit*" (Zeit um, Zeit vorbei,...) angezeigt. Anschließend die Tore verglichen und die Karte des Gewinners eingeblendet.

Wird danach eine *beliebige* Taste gedrückt, beginnt das Spiel von Vorne.

Zeit messen und anzeige

Um die Zeitanzeige kümmern wir uns wieder in der Figur "*Soccer Ball*". Wir benötigen gleich einen Schwarm an Variablen - "*RUNDENDAUER*" ist eine Konstante, welche die Länge des Spiels bestimmt. 90 Sekunden - in Anlehnung an die 90 Minuten eines Fußballspiels - erscheinen angemessen.

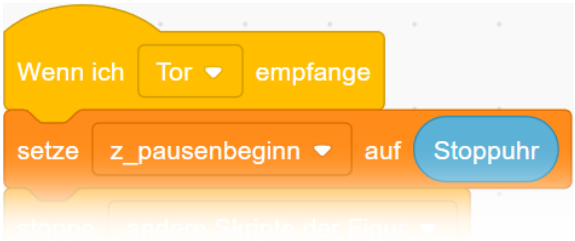
Die Variable "*Zeit*" dient zur Anzeige am unteren Rand des Spielfelds. Stelle sie auf "*groß*" um.

Fünf weitere Variablen sind *nur für die Figur* "*Soccer Ball*" und heißen *z_Minuten*, *z_Sekunden*, *z_Verbleibend*, *z_PausenBeginn*, *z_Pausenzeit*

Innerhalb des *Spielstart*-Skripts ergänze ein Zurücksetzen der *Stoppuhr* und nulle die Variablen *PausenBeginn* und *PausenZeit*.

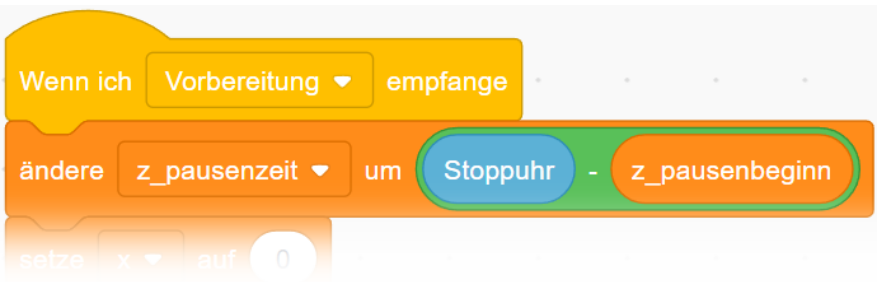
Außerdem muss hier ein Aufruf eines neuen Blocks "*Zeit*" erfolgen.

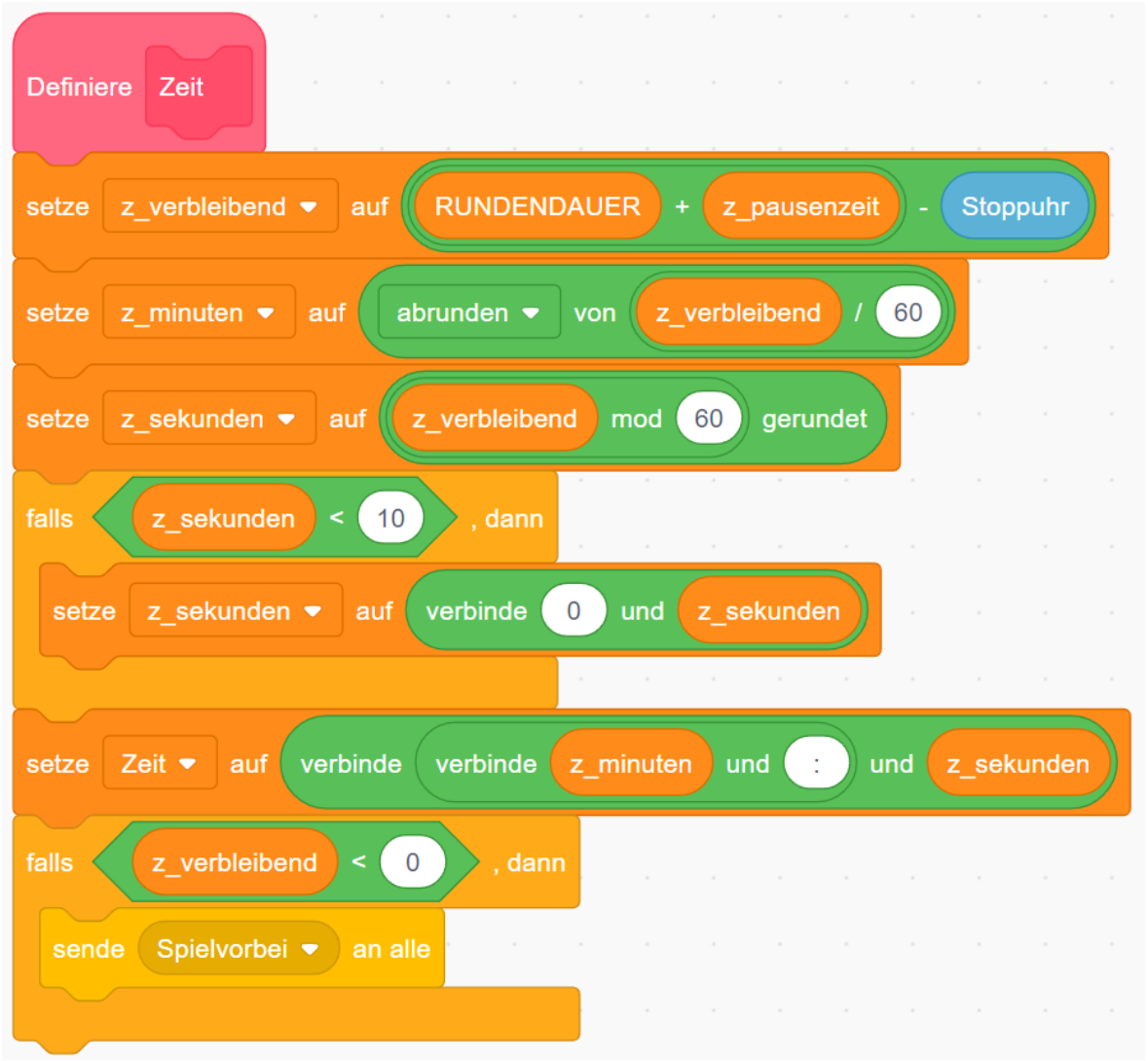
In der *GameLoop* wird ebenfalls der Block "*Zeit*" aufgerufen.



Zu Beginn des *Tor*-Skripts setze den *Pausenbeginn* auf die aktuelle Stoppuhr.

In der *Vorbereitung* ergänze eine Änderung der Pausenzeit um *Stoppuhr - Pausenbeginn.* Pausenzeit wird also um so viel erhöht, wie Zeit zwischen dem Beginn der Pause (weil ein Tor geschossen wurde) und dem Ende der Pause (Vorbereitung auf den nächsten Einwurf) vergangen ist.





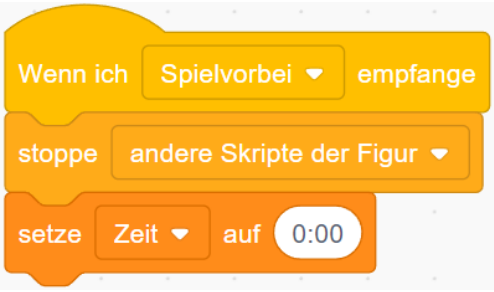
In *Zeit* wird berechnet, was angezeigt werden soll. Die Zeit wird als *verbleibend* auf Null heruntergezählt, dazu wird die *Rundendauer* um die *Stoppuhr* reduziert. Pausenzeiten werden aber nicht berücksichtigt, weshalb sie zur Rundendauer addiert werden.

Um die Verbleibenden Minuten zu erhalten dividieren wir die *Verbleibende* Zeit durch 60. Minutenbruchteile werden durch den *Abrunden*-Block entfernt.

Für die Sekunden verwenden wir *Modulo*. Modulo gibt den Rest einer Division an - in diesem Fall verbleiben die Sekunden. Auch sie müssen *gerundet* werden, da die Stoppuhr auch Sekundenbruchteile zählt. Sekunden müssen immer zweistellig angezeigt werden, bei Werten unter 10 wird deshalb eine führende 0 angefügt.

Die *Zeit* ist die *Verbindung* der *Minuten* und *Sekunden*, mit einem Doppelpunkt dazwischen.

Ist die Zeit abgelaufen - also die *verbleibende* Zeit kleiner Null - endet das Spiel. Die Anzeige des Gewinners wurde schon in den Karten abgehandelt.

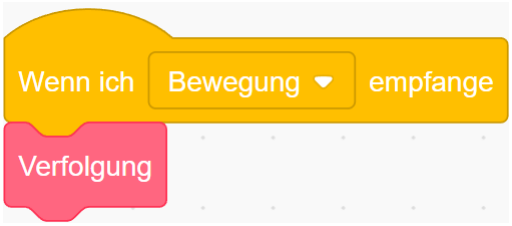


Innerhalb dieser Figur wird ebenfalls auf die Nachricht "*Spielvorbei*" gepasst und alle anderen Skripte unterbrochen. Außerdem setzen wir die *Zeit* auf 0:00, damit es nie so aussieht, als hätten wir zu spät abgebrochen.

Bonus: Schattenwurf

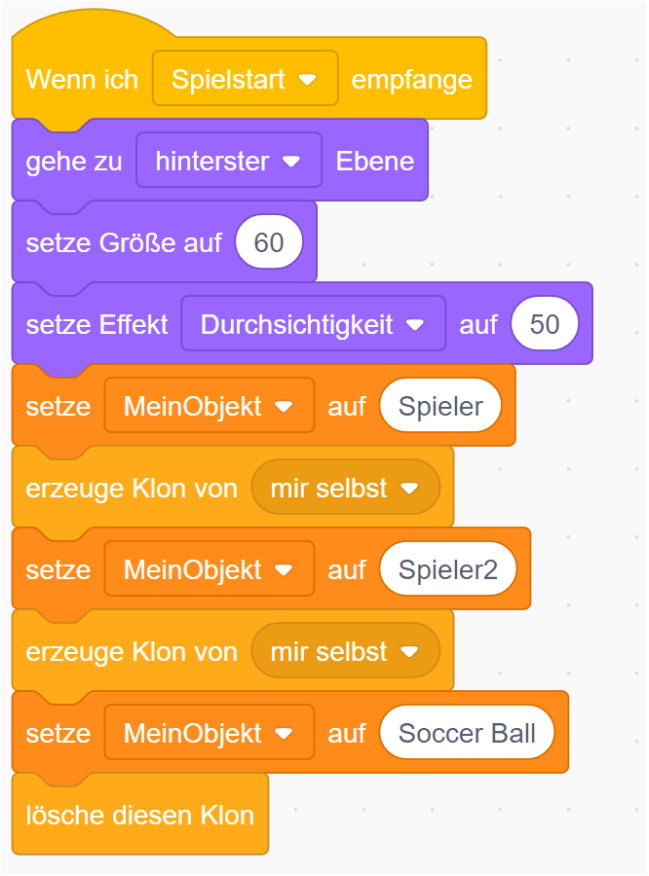
Eine kleiner aber feiner Bonus ist die Ergänzung eines Schattens unter den Figuren und Bällen. Lege eine neue Figur an, welche ebenfalls ein Ball ist. Schnappe dir die obere Kante dieses Balls und verschiebe sie nach unten, bis er ganz flach ist, und färbe ihn schwarz ein.

Diese Figur "*Schatten*" benötigt eine eigene Variable "*MeinObjekt*" *nur für diese Figur*, und einen Block "*Verfolgung*". In diesem bewegt sich der Schatten zum Objekt, ändert aber die y-Position auf *EBENE*.



Immer wenn der Schatten "*Bewegung*" empfängt, wird er die Verfolgung des Objekts aufnehmen.

Bei Empfang der Nachricht "*Spielstart*" geht der Schatten zur *hintersten Ebene*, reduziert seine *Größe auf 60* und wird *halb-Durchsichtig*.
(All das kann auch einmalig eingestellt werden, es ändert sich im Spielverlauf nicht).



Setze nun die Variable "*MeinObjekt*" auf "*Spieler*" und erzeuge einen Klon, dann auf "*Spieler2*" und erzeuge einen weiteren Klon.

Das Original wird zum Schatten des "*Soccer Ball*".

Die Schatten bewegen sich mit dem Ball und den Figuren, bleiben allerdings stets am Boden. Das gibt eine visuelle Stütze, weil es ja gar keinen richtigen Boden gibt, und sieht einfach besser aus.

Fertig?
Schnapp dir einen Freund oder Mentor und fordere ihn zum Fußballduell heraus!
Wer schießt mehr Tore?