Berry Buster – Overview

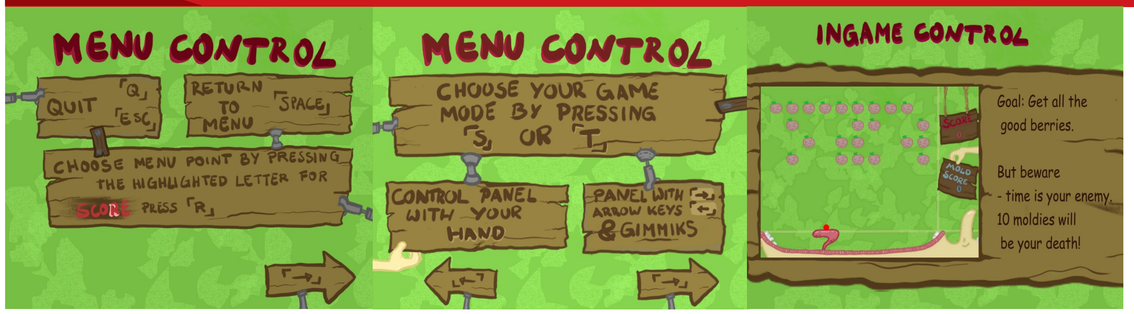
Welcome animation

I. Phase
Berries erstellen & Zeiten und Position des Fallens festlegen
Methode:
- Funktion (True übergeben) zum Fallen auslösen
- For-Schleife: Fallen starten

II. Phase
„Berry Buster“ erscheint Buchstabe für Buchstabe
Methode:
- For-Schleife: passendes Bild updaten (blitten)

III. Phase
“Menu“ erscheint

Tutorial



Methode:
- wenn in gewissem Zeitrahmen geklickt
  → eine Folie vor/zurück
  → Ziel: keine Folie überspringen
- Abruf der Bilder über Listenindex
- wenn letzte Seite erreicht wird → Sprung zu Seite 1
- Indexsetzung über If-Clauses

Menu

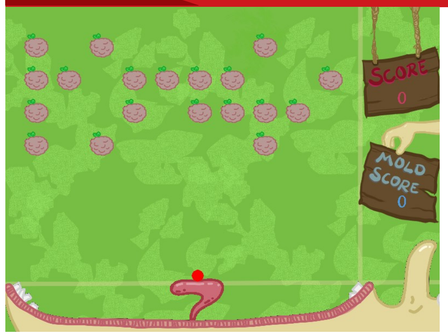
I. Main menu
- Orientierungsscreen

II. Wenn Taste … gedrückt
S - Game Mode 1 starts
T - Game Mode 2 starts
C - Credits
R - Scores
Q - Quit (alternativ ESC)

III. nicht gekennzeichnete Tasten
Tutorial – Tutorial starts
G - secret minigame

Beachte: *Tasten des Numpads wurden passend überklebt, demnach wird im Code z.B. Taste 2 für „S“ genannt.*

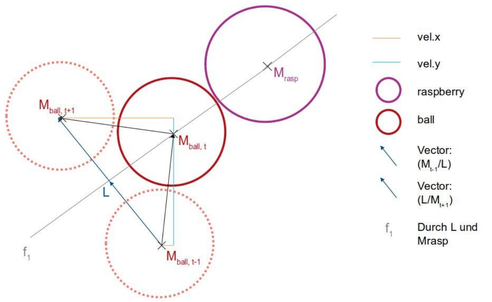
Ingame



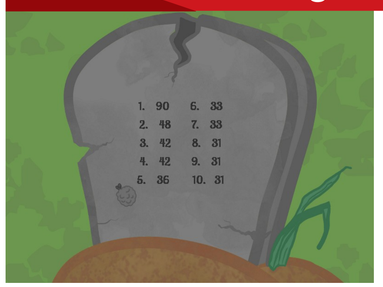
I. Funktion: Starting Phase
- Festlegen randomisierter Beerenposition
- Start Postion Zunge festlegen & mit TAB Ball schießen

II. Funktion: Main game
- Allgemeines Ziel: Beeren in gegebener Zeit einsammeln
  → Start neuer Phase mit Scoreübergabe
  → Reifung definieren und Updaten/Zeichnen aller Objekte
- Tod durch: Zeit abgelaufen, 10 Moldies eingesammelt, Ball verloren

Physik
Panel: Begrenzung durch definierten Bildrand
Ball: Begrenzung Bildrand
- Abprall an Berries (neue Geschwindigkeit/Richtung mittels Punktspiegelung des vorherigen $M_{ball,\ t-1}$ an der Gerade durch M_{rasp} und $M_{ball,\ t}$)



Highscores



I. Start Score = 0
II. Je getroffene Berry: Score += 1
  I. Funktion: Score
  - Vergleicht current score mit Highscoreliste (externe csv)
  - Ersetzt bzw. verschiebt bestehende Werte wenn nötig
  - überschreibt csv
  II. Funktion: Highscore_menu
  - rendered Scores (Top 10)
  - blitted scores auf das Hintergrundbild

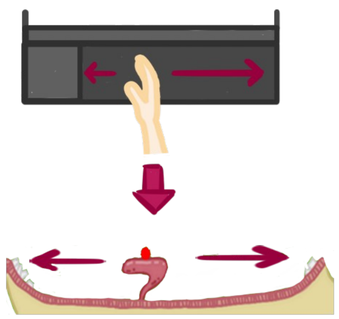
Credits

I. Rendering des Textes
Methode: For-Schleife

II. Blitten des gerenderten Textes
Methode:
- While-Schleife
- Verschieben um 1px/s

III. Aufruf des Menüs nach Ablauf

Ingame: Mode 1



I. Sensornutzung:
Ultrasonic Distanzsensor

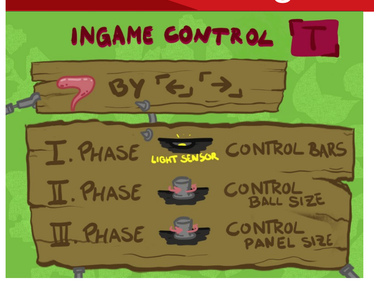
II. Nutzung für Panelbewegung
- Übersetzen von Handposition in Sensorrange → Panelposition ingame (Dreisatz)

III. Problem:
- Unrealistische Ausreißer des Sensors

IV. Lösungsansatz:
- Vor Übersetzung Ausreißer entfernen
- Smoothing über die letzten x-Werte

Methoden:
- Wenn Sensorwert vom arithmetischen Mittel der 4 vorhergehenden um x abweicht → Sensorwert wird nicht in Liste der Werte aufgenommen
- Mitteln der letzten 4 Werte der Liste der Sensorwerte
- Gemittelter Wert wird übersetzt
- Loop: aktueller Sensorwert wird mit diesem neuen Mittel verglichen

Ingame: Mode 2



I. Panelbewegung mittels Pfeiltasten

II. 3 Phasenunterteilung s. Bild

  I. Phase: Lichtsensor lässt Balken erscheinen
  II. Phase: Ballgröße kontinuierlich ändern mittels Rotationssensor
  III.Phase: Panelbreite mittels Rotationssensor stufenweise bestimmen

Methoden:
- Spielphasen definiert per Zeiteinheiten & Aufruf passender Funktionen
  Zeit definiert durch gezählte Frames über pygame.clock.tick()
- Übersetzung des Sensorwertes (Sensoren haben feste Ranges) in nutzbare Werte der zu ändernden Objekte (Bsp.: Ballradius zwischen 2 und 20p)
- Passende LED weisen auf zu nutzende Sensoren hin (analogWrite(led,1))
- Ermöglichung korrekten Ballabpralls durch dynamische Werte (Funktion erkennt bspw. neue Ballgröße oder die Existenz der Balken an) mittels Wertübergabe

PFP | WS1819 | Jenny Iseev & Ulrike Schäfer | *Kontakt:* ulrike.schaefer@campus.tu-berlin.de