|  |  |
| --- | --- |
| Lanemittelpunkte (y-Achse) | Obere Lane: \* 2  Mittlere Lane: \* 4  Untere Lane: \* 6 |
| Lanehöhe | (Obere Lanegrenze – untere Lanegrenze)\*aspect\_ratio\_scaler  aspect ratio scaler, um für andere Seitenverhältnisse als 16:9 eine gleiche Triallänge zu erreichen (ansonsten sind Objekte breiter -> Trial kürzer) |
| Vogelobjekt | Höhe bzw. Breite: |
| Sternobjekt | Höhe bzw. Breite: Lanehöhe |
| Gateobjekt bzw. mittleres Hindernis | Höhe bzw. Breite: Lanehöhe |
| Bewegungsgeschwindigkeit | Pixel pro Frame |
| Sternposition (x-Achse) | Bildschirmbreite \* 1.085  Trial dauert dann ~ 5 Sekunden mit aktueller Bewegungsgeschwindigkeit mit 1 Sekunde Wartezeit vor Trial |
| Mittleres Hindernis Position (x-Achse) | Bildschirmbreit \* 1.085  ~1 Sekunde Zeit zwischen Auftauchen der Sterne auf Bildschirm und Kollision mit mittlerem Hindernis bei aktueller Bewegungsgeschwindigkeit |
| Gate Position (x-Achse) | Bildschirmbreite \* 0.744 |
| Max. Perturbation | Basisverhältnis:  Pixel  Multiplikation mit Faktor 1 aus Exp-Conditions.csv (konstante Schwierigkeit)  Wird alle 100ms auf y-Position des Vogel angewendet.  Perturbation wird ca. alle 250 ms zufällig variiert (maximale Perturbation \* Zufallszahl (uniform) von 0 bis 1) |
| Jump-Threshold | Nach oben:  Nach unten: |
| Vogelorientierung | Lanemittelpunkt (y-Achse) – Vogelmittelpunkt (y-Achse)  Linke obere Ecke des Bildschirms entspricht y-Achse = 0 |
| Gewichtung Vogelorientierung | 0.4575  Wird genutzt um einen modifier-Faktor (siehe unten) zu berechnen. Je kleiner, desto mehr Einfluss hat die Orientierung des Vogels. |
| modifier | Gewichtung \* dependence (1=kompatibel, -1 = inkompatibel, 0=unabhängig) |
| y\_change | Startposition (y-Achse) Cursor – momentane Position (y-Achse) Cursor  Startposition = Bildschirmmittelpunkt  Momentane Position Cursor = y-Achsenposition aktueller Frame (Vor dem Gate gilt stets Startposition = momentane Position; d.h. kein Sprung möglich)  Wenn der modifier-Faktor != 0 (i.e., dependence ist kompatibel oder inkompatibel) ist, wird von y\_change der Quotient:  abgezogen.  y\_change wird in jedem Frame berechnet.  Wenn y\_change den oberen oder unteren Jump-Threshold über bzw. unterschreitet, wird ein Sprung initiiert. |
| Scrollbewegung Mausrad | y-Achsenposition des Vogels wird um  Pixel nach oben bzw. unten verändert |
| Fallevent | Wenn Vogelmittelpunkt (y-Achse) um eine Lanehöhe vom Lanemittelpunkt abweicht, fällt er hinunter.  IF Lanemittelpunkt > Vogelmittelpunkt + Lanehöhe OR Lanemittelpunkt < Vogelmittelpunkt – Lanehöhe THEN bird falls |
| Fehlerevent | Spezifische Fehlernachricht wird für 1.5 Sekunden abgegebildet, dann startet der nächste Trial |
| Updatefrequenz (i.e., FPS) | 100 FPS (Update alle 10ms; falls länger, werden Parameter mit Kompensationsfaktor angepasst und geupdated 🡪 Geschwindigkeit des Experiment unabhängig von Framerate) |

Genaue Position der Objekte, Geschwindigkeit und Thresholds werden internal als floats gespeichert und für Kollisionsberechnung genutzt. Für die Darstellung in pygame werden die floats in integer umgewandelt. Trotz teilweise unterschiedlicher Größe (bedingt durch reward size) haben Sterne die gleichen Kollisionsausmaße.