Computer Games Exercises: 2024s s06 (all)

https://www.umm.uni-heidelberg.de/miism/

Contents

Answer header

Please put the author information in the header of all code files.

- `name (Name)`
- `coauthor list`

C03b: Collision time

Preparation

Please read the source code of the "`Collision time`" game and understand the game logic.

There is a path along which an object is moving from left to right, and a wall. The object movement is identified by the offset along the path with the speed V. After every interval T, the collision detection is performed. When a collision is detected at T_2, the collision time is calculated with the information at $T_1 (= T_2 - T)$ and T_2.

Task

Please extend the game.

- Use the parameter value of speed V and interval T from the interface to move the object, and show the results in the corresponding labels when the object stops moving.
- Implement the function `distance()` to calculate the physical distance from the object to the wall.

$$d(x_{\text{object}}) = \begin{cases} x_{\text{wall\_left}} - x_{\text{object}} & \text{if } x_{\text{object}} <= x_{\text{wall\_middle}} \\ x_{\text{object}} - x_{\text{wall\_right}} & \text{if } x_{\text{object}} > x_{\text{wall\_middle}} \end{cases}$$

- Update the function `_process()` to perform collision detection after every interval T.
  - When the distance from the object to the wall is negative, there is an intersection.
  - When there is no collision (NC), draw a red circle at the current position of the object (using `draw_circle()`), and keep the object moving.
  - When there is an intersection (IS), stop the object, estimate the collision time using the "fast correction method" and the "bisection method", and show the results in the corresponding labels.
  - The collision time is represented in seconds with 3 digits right to the decimal, referring to the start time of the object movement `timeStart`.
- Implement the function `fast_correction()` to estimate the collision time using the "fast correction method" at time T_2.

$$\tau = T_1 + d(x(T_1))/V$$

- Implement the function `bisection()` to estimate the collision time using the "bisection method" at time T_2.

```
$t_1 = T_1, t_2 = T_2$
while $|t_1 - t_2| > T/1000$ :
    $\tilde{t} = (t_1 + t_2)/2$
    if NC at $\tilde{t}$ :  $t_1 = \tilde{t}$
    else :  $t_2 = \tilde{t}$
$\tau = t_1$
```

Note: $d(x(\tilde{t}))$ should be explicitly calculated by moving the object along the path to perform collision detection at time $\tilde{t}$.

- Update the function `_process()` to call the function `record_actual()`.
  - Implement the function `record_actual()` to perform collision detection at each frame, and record the actual collision time by the frame time.
  - The actual collision time is represented in seconds with 3 digits right to the decimal, referring to the start time of the object movement `timeStart`.

Questions

Write the corresponding answers in the script file.

- With which interval T the collision may not be detected when speed $V = 2000$ pixel/s?
- When the object may be overseen, which strategy has to be used instead?
- In the lecture you learned that we have a broad and narrow phase: Could you improve the collision time estimate via bisection using these two phases? What is the expected gain in performance on average?