Computer Games Exercises: 2024s s09 (all)

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

Contents

Answer header

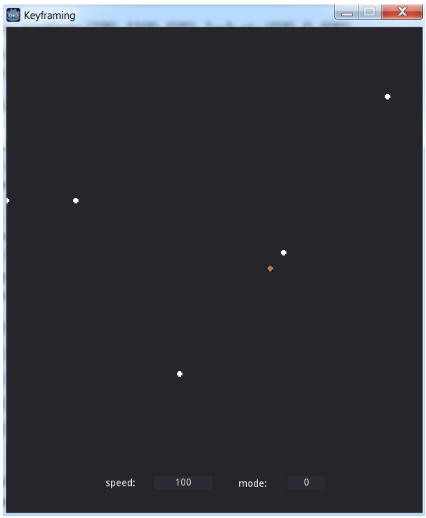
Please put the author information in the header of all code files.

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

A03: Keyframing

Please read the material of the "`Keyframing`" game. In this task, you should implement a keyframing method, to move a ball along a spline with some predefined key frames.

Scene



Create a scene with the following items:

Object	Type	Position	Size	Texture
`panel1`	Control:Panel	(0, 0)	(600, 600)	
`sphere1`	Node2D:Sprite2D	(0, 250)		sphere.png
`sphere2`	Node2D:Sprite2D	(100, 250)		sphere.png
`sphere3`	Node2D:Sprite2D	(250, 500)		sphere.png
`sphere4`	Node2D:Sprite2D	(400, 325)		sphere.png
`sphere5`	Node2D:Sprite2D	(550, 100)		sphere.png
`ball`	Node2D:Sprite2D	(0, 250)		ball.png
`panel2`	Control:Panel	(0, 600)	(600, 100)	

- Add several Control:* objects to `panel2`, which should be used to control the speed and movement mode and show text.

Variables

Create a script for the ball with the following variables:

- an array to store the location of waypoints represented by the spheres
- an array for the derivatives at the waypoints, which contains the following items

```
(2, 0)
(1, 2)
(2, 1)
(1, -2)
(2, 0)
```

- a variable of speed v

Methods

Implement the following methods:

- **Hermite Spline** defines the path with the waypoints and derivatives:

$$\begin{aligned} c(t) &= h_{00} \cdot p_0 + h_{10} \cdot p_1 + h_{01} \cdot d_0 + h_{11} \cdot d_1 \\ h_{00} &= 2 \cdot t^3 - 3 \cdot t^2 + 1 \\ h_{10} &= -2 \cdot t^3 + 3 \cdot t^2 \\ h_{01} &= t^3 - 2 \cdot t^2 + t \\ h_{11} &= t^3 - t^2 \end{aligned} \tag{1}$$

  where p_0, p_1 are two neighboring waypoints, d_0, d_1 are their derivatives and $t \in [0, 1]$ is the parameter.

- **Catmull-Rom Spline** estimates the derivatives for the inner points by:

$$d_k = \delta_k \cdot (p_{k+1} - p_{k-1}). \tag{2}$$

  With $\delta_k = 0.5$ and $d_{\text{first}} = d_{\text{last}} = (10, 0)$, the path can be calculated as the Hermite Spline.

- Implement the **arc length calculation** of the parametric functions.
- Control the movement mode by one Control:* object in `panel2`. It switches between Hermite Spline and Catmull-Rom Spline.
- Control the speed by another Control:* object in `panel2`. The speed value is used to control the movement.
- Method `move()` moves the ball for each frame according to the control.
  - The piece moving time T_{piece} between two neighboring waypoints should be calculated by their arc length and the input speed: $T_{\text{piece}} = l_{\text{piece}}/v$.
  - The parameter t should be linearly interpolated by the frame time T_{frame} and the piece moving time T_{piece}: $t = (T_{\text{frame}} - T_{p0})/T_{\text{piece}}$.
  - When the parameter $t > 1$, the ball moves to the next piece.
  - The outcome of the game should be a smooth movement of the ball along a piecewise polynomial path from `sphere1` to `sphere5`.
  - When the ball reaches `sphere5`, it starts from `sphere1` again.
  - Display the arc length of the path from `sphere1` to the current ball position dynamically in another Control:* object in `panel2`.

Questions

Write the corresponding answers in the script file.

- The extension of keyframing to a deformable object is realized via free-form deformation. In contrast, for physics-based animation, the procedure is automatic and completely defined by the applied forces to the object under investigation. How can keyframing be transferred to physics-based animation henceforth?