Animation and Movie Making

Intended Learning Outcomes

- Distinguish two types of animation
- Describe the four steps of animation
- Describe key frame and intermediate frame generation techniques
- Able to model and program common animation effects such as acceleration, deceleration, and periodic motion

Two Types of Animation

- Real time animation
  - Update parts of image in real time as soon as available
- Frame by frame animation
  - Use two frame buffers
  - Display first buffer content
  - Update on the second buffer
  - Switch the two buffers when the new image has finished drawing on the second buffer
  - Use in system that does not require real time e.g. movie production

Comparisons

- Real time animation
  - Adv. Critical Information displayed as soon as available
  - Disadv. Refresh rate of each pixel must be at least 16 frames/sec to avoid flickering
  - Used in real time systems e.g. flight simulator, multi-player games

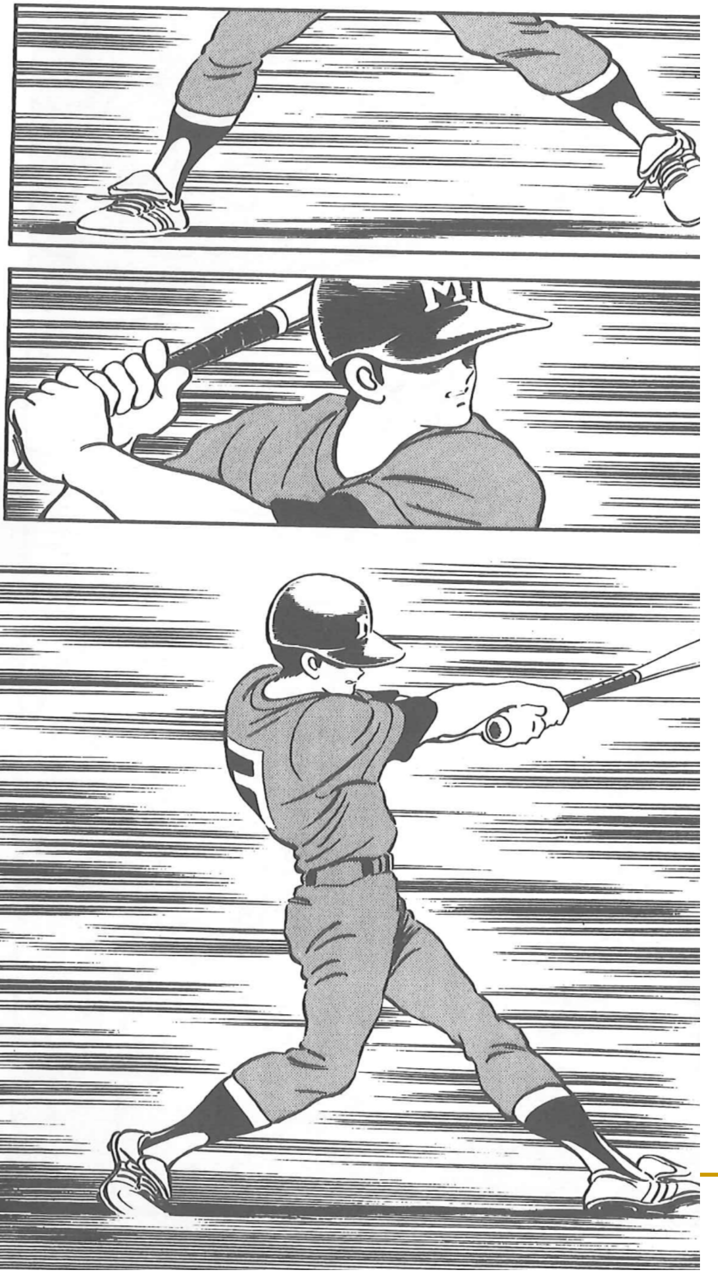
- Frame by frame animation
  - Adv. No flickering even if the refresh rate is low
  - Disadv. Display of information may be delayed up to one frame
  - Used in non-real time systems e.g. movie

Designing an Animation

- Story Board
  - outline of the action. Defines the motion sequence as a set of basic events that are to take place
- Object Definitions
  - choose the object representation and movement of each object in the story
- Generation of Key Frames
  - generate a detailed image of the scene at a certain time in the animation sequence
  - More key frames are specified when the motion is intricate
- Generation of In-between Frames
  - Intermediate frames between the key frames.
  - The number of in-betweens needed is determined by the media to be used to display the animation.

Key frames

From comic “H2”



Generation of in-between frames from key frames

- Key frames can be generated by the CG pipeline
- Morphing can be used to generate in-between frames
- Morphing – short form for metamorphosis
- It is a transformation of object shape from one form to another

Morphing

- Step 1 : Equalize the number of vertices of the two shapes
- Step 2 : Find correspondence between each pair of vertices
- Step 3 : Find intermediate positions of the vertices by interpolation

Algorithm

Input : Key frames k and k+1

Algorithm

1. Let V_k be the number of vertices in key frame k. Compute

$$V_{\max} = \max(V_k, V_{k+1}) \qquad V_{\min} = \min(V_k, V_{k+1})$$

and

$$N_{ls} = (V_{\max} - 1) \bmod (V_{\min} - 1)$$

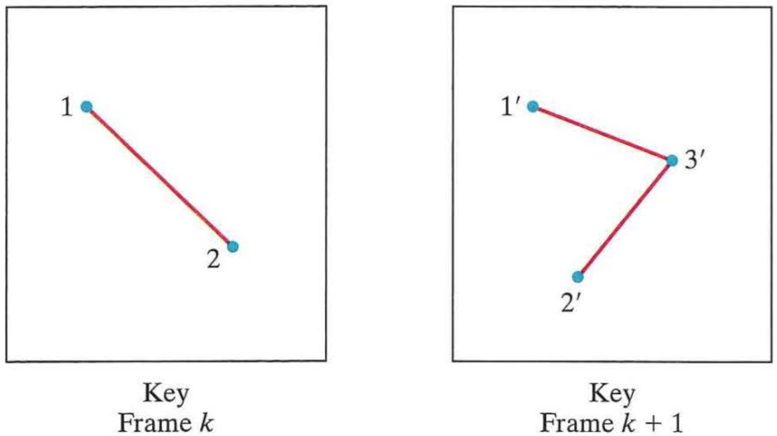
$$N_p = \mathrm{int}\left(\frac{V_{\max} - 1}{V_{\min} - 1}\right)$$ // int (x) takes the largest integer smaller than x

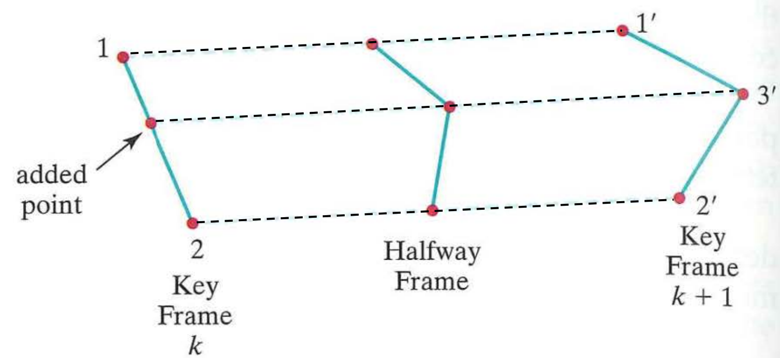
2. Add N_p points to N_{ls} line sections of $\text{keyframe}_{\min}$ (the key frame with less number of vertices)
   Add N_p - 1 points to the remaining edges of $\text{keyframe}_{\min}$

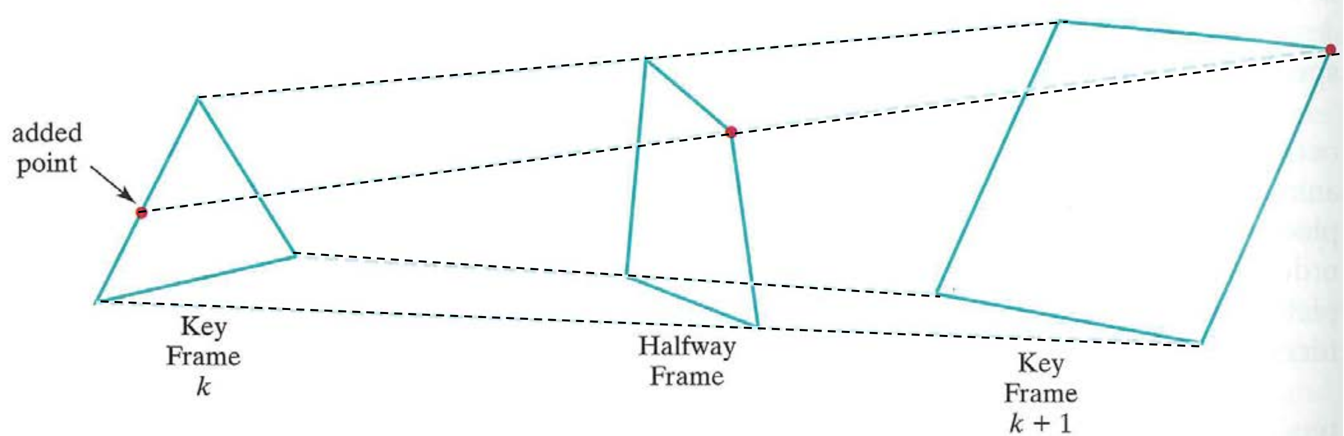
   // now both key frames have equal number of vertices

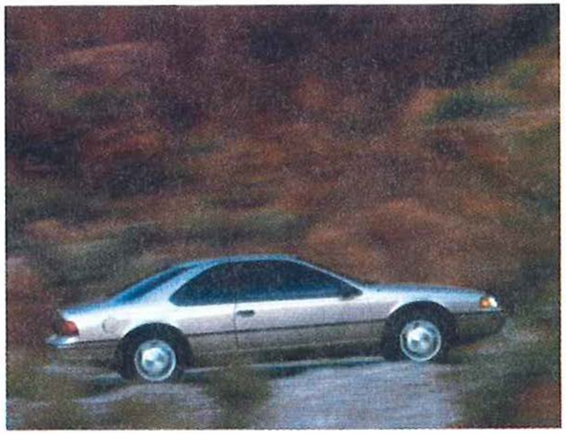
3. Linearly interpolate for each pair of corresponding vertices in the two key frames to generate the in-between frames

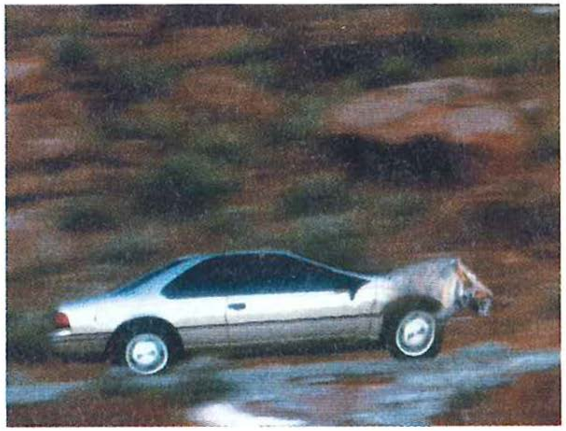
Output: a set of in-between frames


1
2
Key
Frame k
1′
3′
2′
Key
Frame $k + 1$


1
1′
3′
added
point
2
2′
Key
Frame
k
Halfway
Frame
Key
Frame
$k + 1$


added
point
Key
Frame
k
Halfway
Frame
Key
Frame
$k + 1$





(a) (b)

(c) (d)

Simulating Acceleration and Deceleration

- Idea : Adjust the time spacing of successive frames
- n in-between frames for two key frames at t = t_1 and t_2
- Constant velocity

$$tB_j = t_1 + \frac{j\Delta t}{n+1} \qquad \mathrm{j} = 1, 2, \ldots, \mathrm{n} \qquad \Delta t = t_2 - t_1$$

Empirical functions

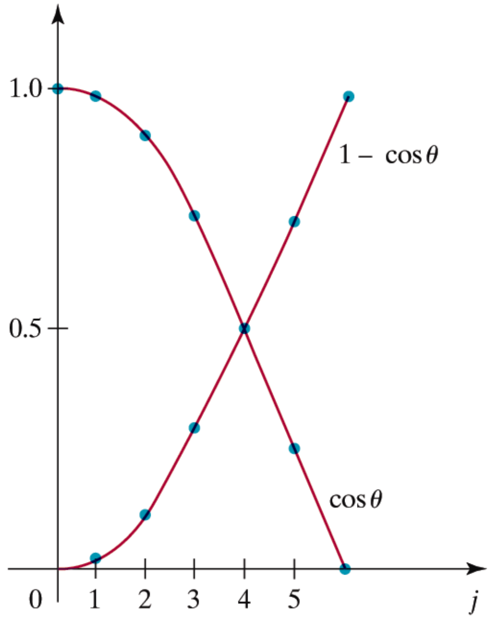
- Acceleration: Use empirical function 1- cos θ   0< θ < π/2

$$tB_j = t_1 + \Delta t[1 - \cos\frac{j\pi}{2(n+1)}]$$

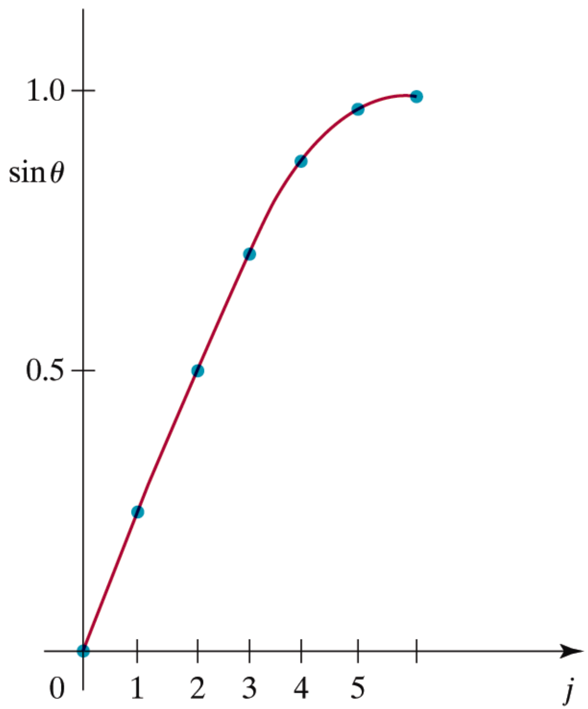
- Deceleration: Use sin θ

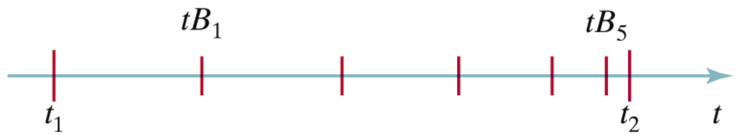
$$tB_j = t_1 + \Delta t[\sin\frac{j\pi}{2(n+1)}]$$

- Accelerate then decelerate: Use $\frac{1}{2}(1-\cos\theta) \qquad 0 < \theta < \pi$

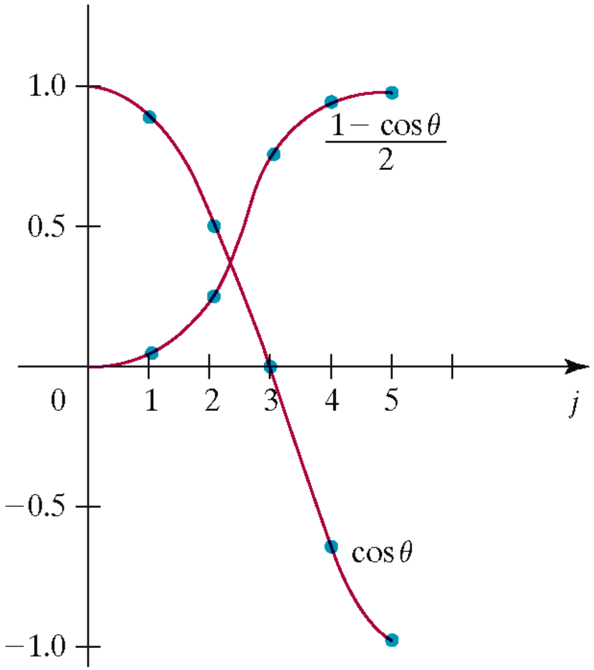


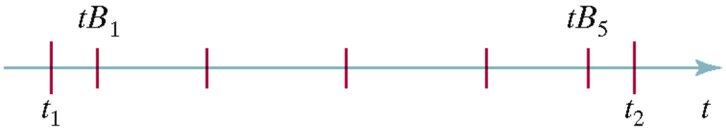
Acceleration





Deceleration



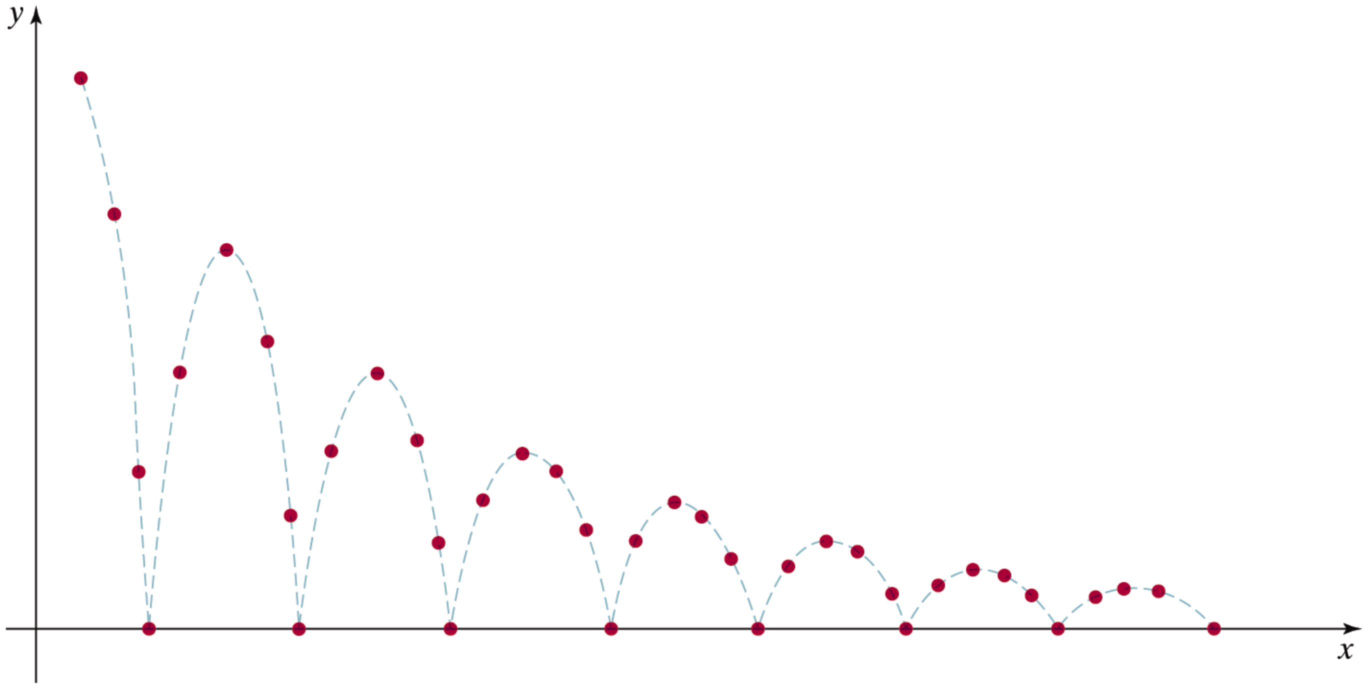


Acceleration then Deceleration

Specifying Motion (1)

- For general motions, empirical functions are not accurate enough
- Three ways to calculate motion
  - Direct Motion specification
    - Solve the motion equations, then just plot the trajectory
    - Example: simple harmonic motion
  - Kinematics and dynamics
    - Kinematics :   calculate position, velocity and acceleration

$$v = u + at \qquad s = s_0 + ut + \frac{1}{2}at^2$$



Simple harmonic motion

Specifying Motion (2)

- Inverse Kinematics

  Specify the initial and final conditions, then the system solves for the motion
- Dynamics

  Specify the forces : Physically based modelling

$$F - kv - h(x - x_0) = ma$$

- Inverse Dynamics

- Goal Directed System
  - Specify desired behaviour : “Walk”, “Run”
  - Converted into mathematical motion by the system

Periodic Motion

- Motion must be synchronized with the frame rate, otherwise may result in incorrect motion
- A typical example is shown in the figures below.
- Solutions
  - Generate a frame after each fixed angle increment, but this may cause other problems if the periodic motion is too fast
  - Use timer and ask user to have a certain minimum graphics capability in their computer (common practice in games)
  - Periodically reset parameters to prevent numerical error build up

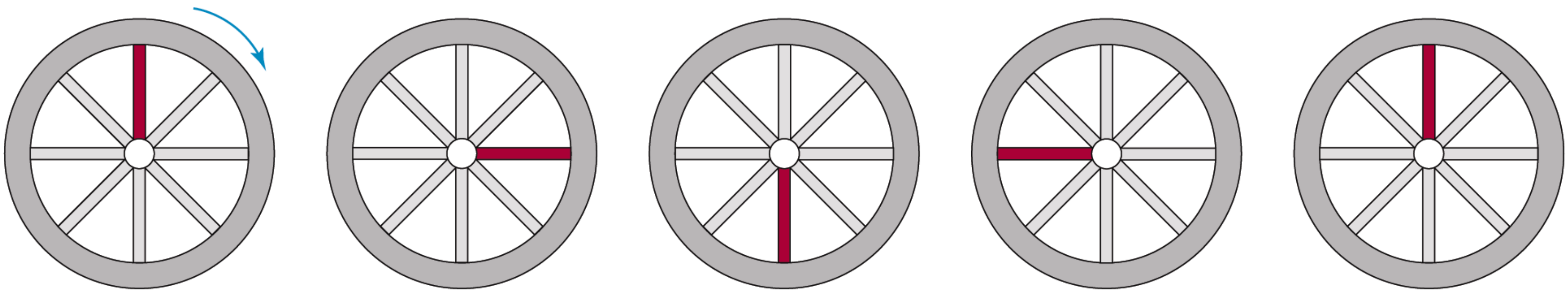
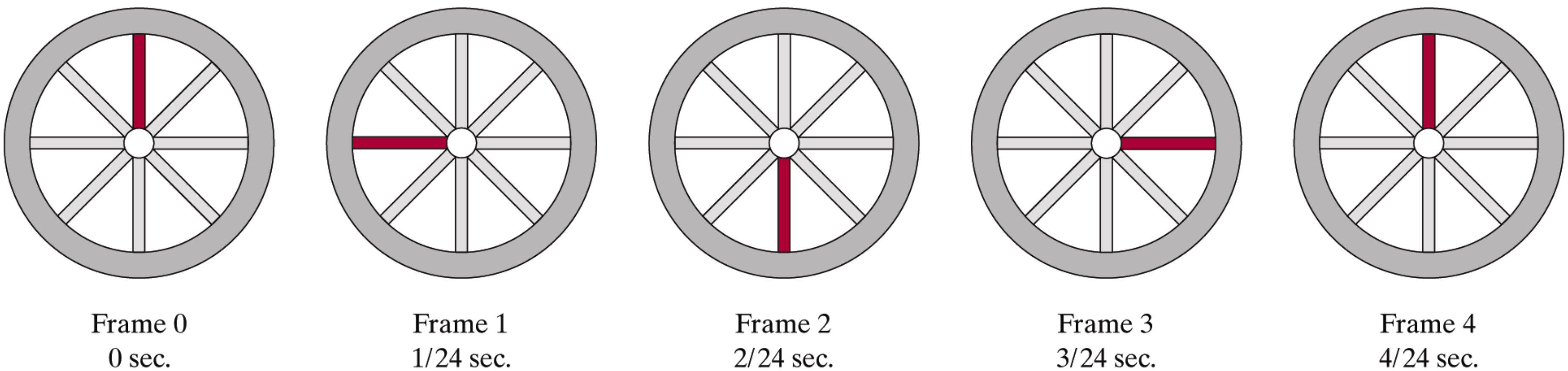


Figure 13-21

Five positions for a red spoke during one cycle of a wheel motion that is turning at the rate of 18 revolutions per second.



The first five film frames of the rotating wheel in Fig. 13-21 produced at the rate of 24 frames per second.

OpenGL Commands

- Double Buffering
  - *glutInitDisplayMode (GLUT_DOUBLE)*
  - *glutSwapBuffers ( );*

- To produce an animation
  - *glutIdleFunc (animationFcn)*
  - *animationFcn* is a procedure written by the user to update the animation parameters
  - *glutPostRedisplay ( );*

- See example program in pg. 410

- Using the timer
  - *glutGet(GLUT_ELAPSED_TIME)*

References

- Text: Ch. 12