Computer Graphics

Unit 6

- **Segment and Animation**

Contents

- **Segment:** Introduction, Segment table, Segment creation, closing, deleting and renaming, Visibility.
- **Animation:** Introduction, Design of animation sequences, Animation languages, Key-frame, Morphing, Motion specification.
- **Colour models and applications:** Properties of Light, CIE chromaticity Diagram, RGB, HSV, CMY, YIQ, colour Selection and applications.

Segments

- In general, the image on display screen is often composed of pictures of several objects.
- Each object has a set of attributes such as size, colour, position etc.
- We might wish to see all these objects simulteneously or a single object at a time. For this we need to organize the image information in a particular manner.
- The image information is stored in Display file.
- A display list (or display file) is a series of graphics commands that define an output image. The image is created (rendered) by executing the commands.
- Existing structure of display file does not satisfy the requirements of viewing image. Hence the display structure is modified to reflect the sub picture structure. To achieve this display file is divided into **segments.**

- Each segment corresponds to a component or a object of the overall display and is associated with a set of attributes.
- Along with the attributes, the segment is also associated with the image transformation parameters such as scaling and translation along X and Y direction, rotation and shearing.
- Thus, segment allows:

➔ Subdivision of the picture

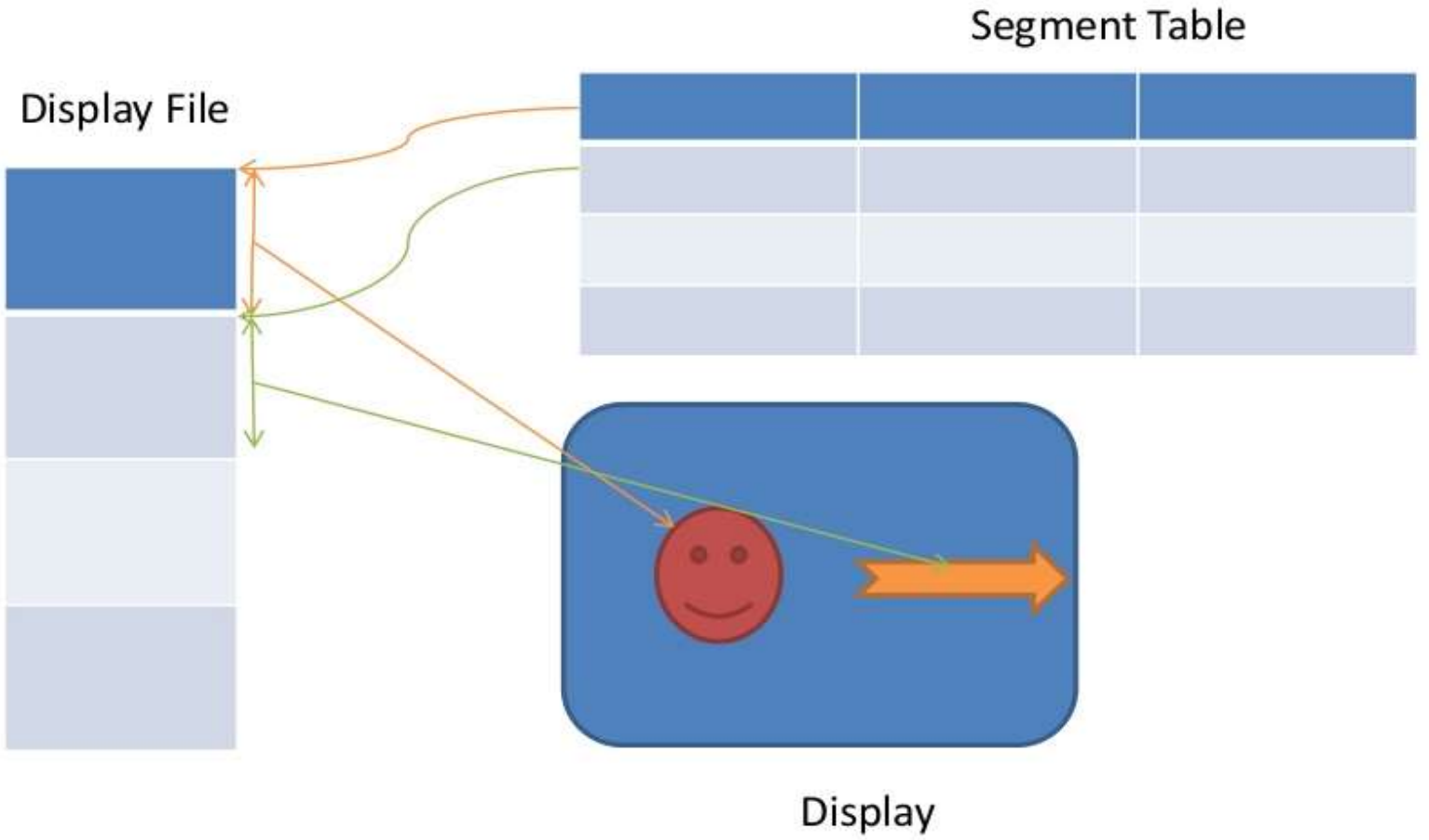
➔ Visualization of a particular part of the picture

➔ Scaling, rotation, translation and shearing of a particular object in the picture

- The structure used to organize all this information related to segments is called **segment table.**

Segment Table

The Segment table indicates the portion of the display file used to construct the picture



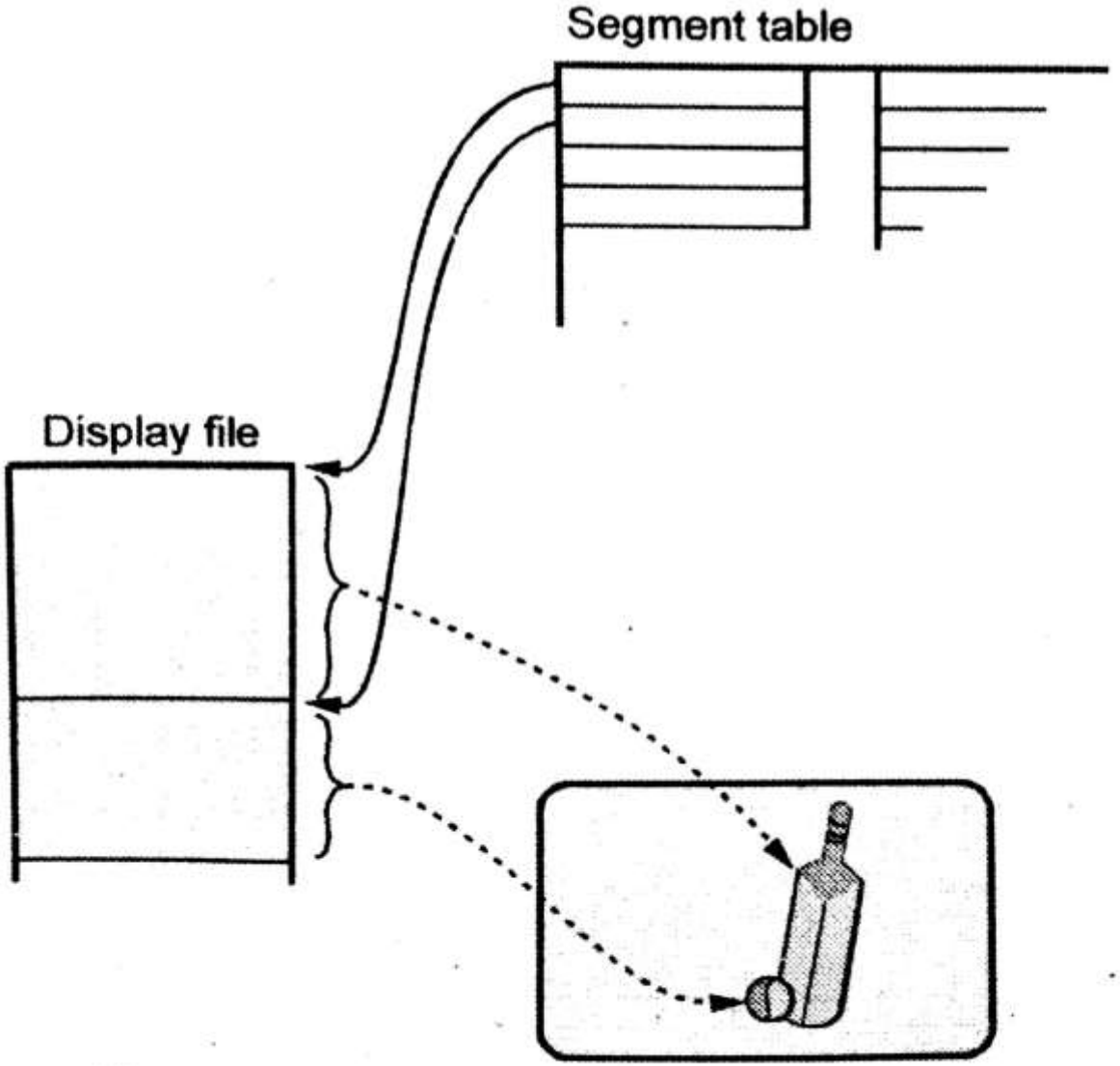


Fig. 8.2.1 Segment table and display file

- Each row in the segment table represents information of one segment including its name, position, size, visibility, attributes and image transformation parameters.
- If we wish to make segment 4 visible then this is achieved by setting the visibility entry for segment 4 to 'ON'.
- The display file interpreter initially checks the start, size and visible attribute of the segments and it interprets only those segments which are to be made visible.

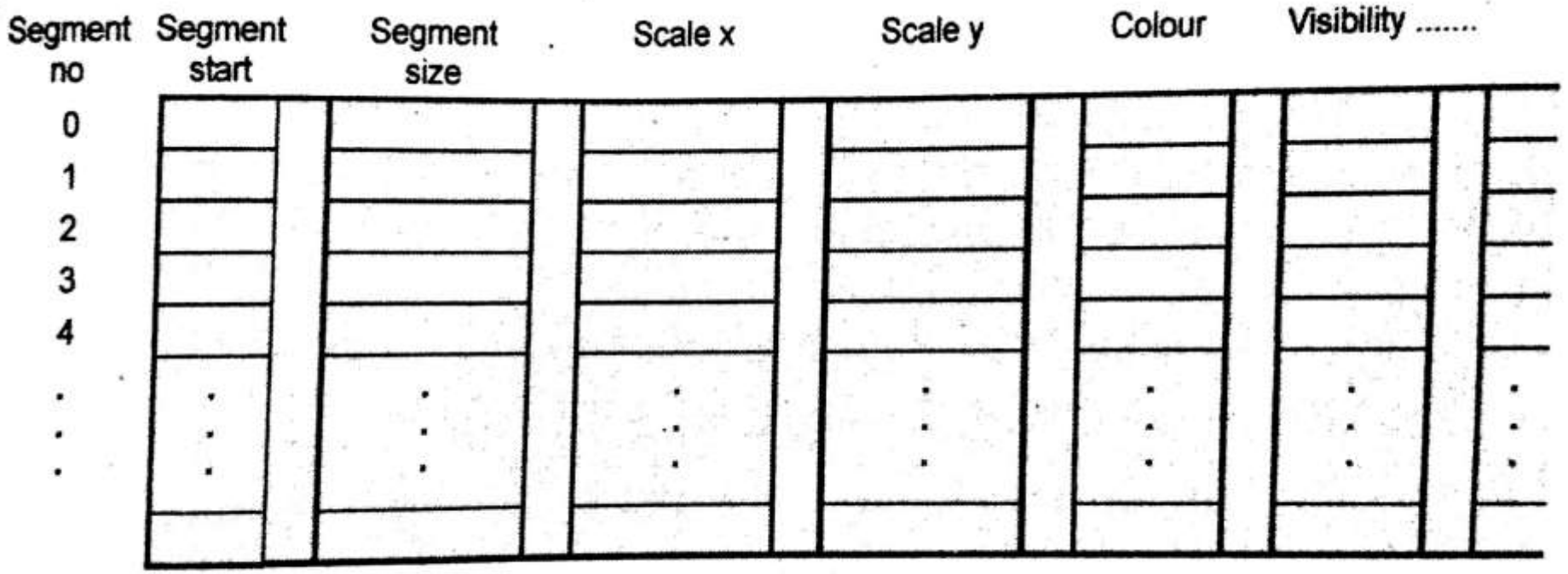


Fig. 8.2.2 Segment table

Segment table can be illustrated as below mentioned table:

Segment Number	Start Position	Size	Visibility	tx	ty	sx	sy	θ
1	1	15	ON	10	5	2	0.5	15
2	16	10	OFF	5	15	0.5	1	45
3	27	20	ON	4	11	1	3	90
4	48	14	ON	7	3	2	1	40

- The segment table is formed by using arrays. An alternative approach is linked list.
- In case of arrays, maximum no. of segments that can be included in the segment table are equal to the length of the arrays. But with linked list there is no such limit on the maximum no. of segments; as list is dynamic.
- In linked List, ordering is achieved by simply adjusting the links. In case of arrays we have to move actual segment information in the process of sorting the segments.
- The Disadvantage of linked list is that it requires more storage to hold the links.

Segment number	Segment start	Segment size	Scale x	Scale y	Colour	Visibility	Link
1							3
2							4
3							2
4							5
5							Null /

Fig. 8.2.3 Segment table using linked list

Segment Creation

- We Must give the segment name so that we can identify it. For example say that there is segment no. 3, then all following commands would belong to segment 3. We could then close segment 3 and open another.
- First thing to create a segment is to check whether some other segment is open. We can not open two segments at the same time because we would not know to which segment we should assign the drawing instructions. If there is segment open, we have an error
- Next we should select valid segment name and check whether there already exists a segment under this name. If so, we again have an error.
- The first instruction belonging to this segment will be located at the next free storage area in the display file.
- The current size of the segment is zero since we have not actually entered any instructions into it yet.
- The attributes of the segment are initialized as a default attribute values.

Algorithm: Create Segment

Algorithm : Create Segment

1. Check whether any segment is open; if so display error message : "Segment is still open" and go to step 9.
2. Read the name of new segment.
3. Check whether the new segment name is valid; if not display error message : "Not a valid segment name"and go to step 9.
4. Check whether the new segment name is already existing in the same-name list; if so display error message : "Segment name already exists" and go to step 9.
5. Initialize the start of the segment at the next free storage area in the display file.
6. Initialize size of this segment equal to zero.
7. Initialize all attributes of segment to their default values.
8. Indicate that the new segment is now open.
9. Stop.

- A segment is created by the function:
- **CREATE SEGMENT(ID)**
- where ID is the name by which the segment is to be known.
- This segment is then the open segment.
- BEGIN
- SEGMENT-START[SEGMENT-NAME] <- FREE;
- SEGMENT-SIZE[SEGMENT-NAME] <- 0;
- VISIBILITY[SEGMENT-NAME] <- VISIBILITY[0];
- ANGLE[SEGMENT-NAME] <- ANGLE[0];
- SCALE-X[SEGMENT-NAME] <- SCALE-X[0];
- SCALE-Y[SEGMENT-NAME] <- SCALE-Y[0];
- TRANSLATE-X[SEGMENT-NAME] <- TRANSLATE-X[0];
- TRANSLATE-Y[SEGMENT-NAME] <- TRANSLATE-Y[0];
- END;

- It is important to note that at any time, only one segment can be open, thus a series of function calls such as:

- CREATE SEGMENT(1)
- output primitives

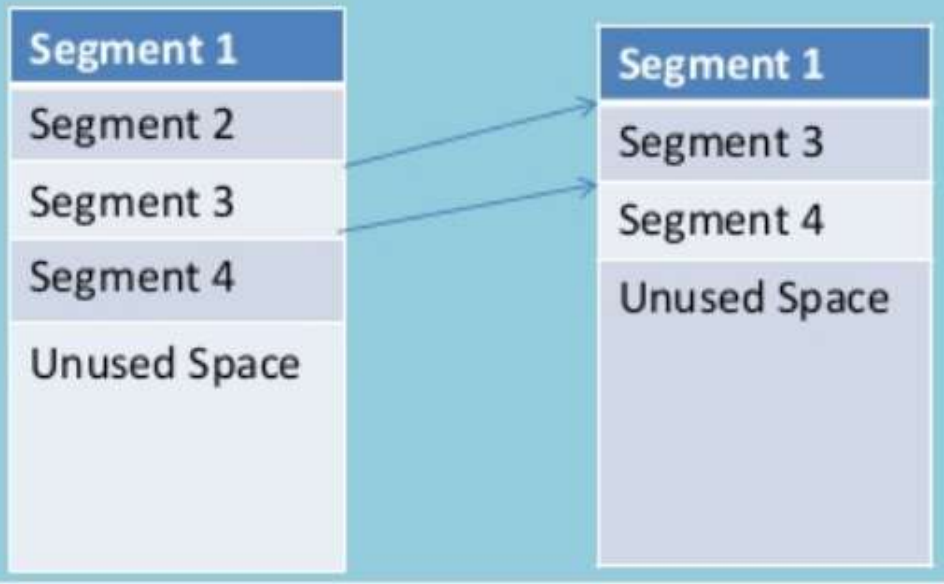
  - CREATE SEGMENT(2) **ILLEGAL!!**
- output primitives
- CLOSE SEGMENT
- CLOSE SEGMENT

Closing a Segment

- Once segment is open, we can enter display file instructions in it. After completion of entering all display file instruction, the segment must be closed.
-
  - CREATE SEGMENT(1)
  - output primitives
  - CLOSE SEGMENT

Deleting a Segment

- When we want to delete a particular segment from the display file then we must recover the storage space occupied by its instructions and make this space free for some other segment. To do this we must not destroy and re-form the entire display file, but we must delete just one segment, while preserving the rest of the display file.
- Here we have used arrays to store the display file information.



Algorithm: Deleting a Segment

1. Read the name of the segment which is to be deleted.
2. Check whether the segment name is valid; if not display error "Segment not valid" go to step 8
3. Check whether the segment is open, if yes, display error message " Can't delete open segment" go to step 8.
4. Check whether the size of segment is greater than 0, if no, no processing is required as segment contains no instructions.
5. Shift the display file elements (other segments in display file) which follow the segment which is to be deleted by it's size.
6. Adjust the starting positions of the shifted segments by subtracting the size of the deleted segment from it.
7. Stop.

Renaming a Segment

- The display processor is continuously reading the display file and showing its contents.
- Suppose we wish to use this device to show an animated character moving on the display.
- To display a new image in the sequence we have to delete the current segment and re-create it with the altered character. The problem in this process is that during the time after the first image is deleted and time before the second image is completely entered, only a partially completed character is displayed on the screen.
- We avoid this problem by keeping the next image ready in the display file before deleting the current segment.

- Segment which is to be deleted and segment which is to be replaced with must exist in display file at the same time.
- We do this by building the new invisible image (making visibility of that image as OFF) under some temporary segment name. When it is completed, we can delete the original image, make the replacement image visible, and rename the new segment to become the old segment to achieve apparent motion.
- The idea of maintaining two images, one to show and one to build or alter, is called **double buffering.**

Algorithm: Renaming a Segment

1. Check whether both old and new segment names are valid; if not display error message "Not valid segment names" and go to step 6.
2. Check whether any of the two segments are open. If open, display error message "Segment still open" and go to step 6.
3. Check whether the new name we are going to give to the old segment is not already existing in the display file. If yes, display error message "Segment already exists" and go to step 6.
4. Copy the old segment table entry into new position.
5. Delete the old segment.
6. Stop.

Visibility

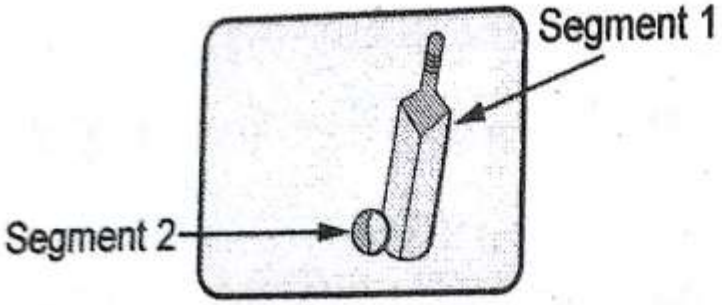
-Each Segment is given a visibility attribute.

-The segment's visibility is stored in an array as part of the segment table.

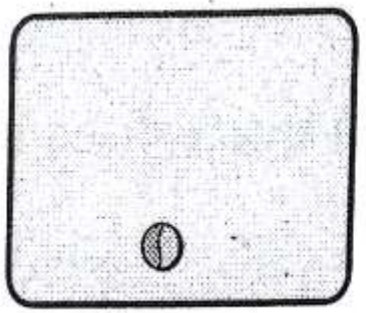
-By checking this array we can determine whether or not the segment should be displayed.

Image Transformation

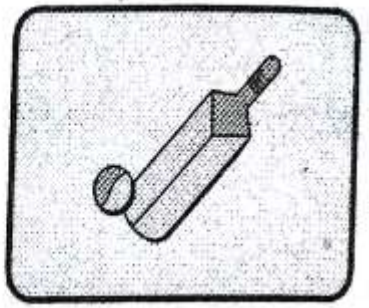
- One of the major advantage of computer graphics is that we can display pictures with certain alterations if required. For example, we can see image from a different angle, we can zoom particular part of the image, we can change the size of the image, we can change the position of particular part of the image.
- These changes are easy to perform because the graphics image information is stored in different segments and we can apply changes to different segments independently.
- The changes in the original graphics information can be done by performing mathematical operations which are called **transformations.**



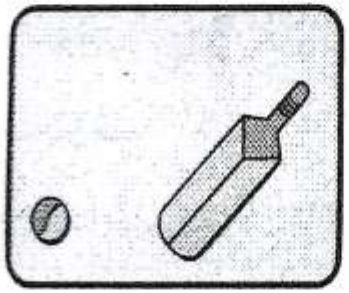
a) Original image



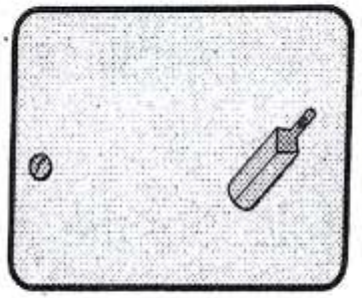
b) Segment 1-visibility OFF



c) Segment 1-rotation 45°



d) Segment rotation 45° and segment 2 translate x



e) Segment 1- rotation 45° scale ¼ Segment 2- scale ¼, translate x

f) Segment 1-rotation 45° scale ¼ Segment 2- visibility OFF

Algorithm : Setting Image Transformation

1. Read the name of segment, say s_name.
2. Check whether the name is valid or not. If not display error message " Not a valid segment " and go to step 8.
3. Read translation factor, say t_x and t_y.
4. Read the scaling factors, say s_x and s_y.
5. Read the angle of rotation, say θ.
6. Set various transformation parameters for the segment s_name.

   Translate_x [s_name] = t_x.

   Translate_y [s_name] = t_y.

   Scale_x [s_name] = s_x.

   Scale_y [s_name] = s_y.

   Rotate [s_name] = θ.
7. Check the visibility of the segment s_name. If visible then create a new frame for it.
8. Stop.

Animation

- **To animate** means "to give life to“. Animating is moving something which can't move itself.
- **Computer animation** is the use of computer graphics to create animations. It is a subfield of computer graphics and animation.
- **Animators** are artists who specialize in the creation of animation.
- An animator's job is to take a static image or object and literally bring it to life by giving it movement.
- In computer animation, animators use software to draw, model and animate the objects and characters.

- The basic idea behind animation is to playback the recorded images at the rates fast enough to fool the human eye into interpreting them as continuous motion.
- Animation can make a series of dead images come alive.
- Animation can be used in many areas like entertainment, computer aided-design, scientific visualization, training, education, e-commerce, and computer art.

Design of Animation Sequences

- Common Steps of designing the animation sequence are as given:

  1. Storyboard layout

  2. Object definitions

  3. Key-frame specifications

  4. Generation of in-between frames

1. Storyboard layout:

- The Storyboard layout is an outline of the action. It defines the motion sequence as a set of basic events that are to take place. Such ordered set of events gives the motion sequence.
- The storyboard could consist of a set of rough sketches or it could be a list of the basic ideas for the motion.

- The storyboarding process was developed at Walt Disney Productions during the early 1930s, after several years of similar processes being in use at Walt Disney and other animation studios.

2. Object definitions:

- Each active section of the scene is treated as an object.
- The object definition is specified for all participant objects in action.
- Along with objects, the description of the movements that are to be performed by each character or object is also specified.

3. Key-frame specifications:

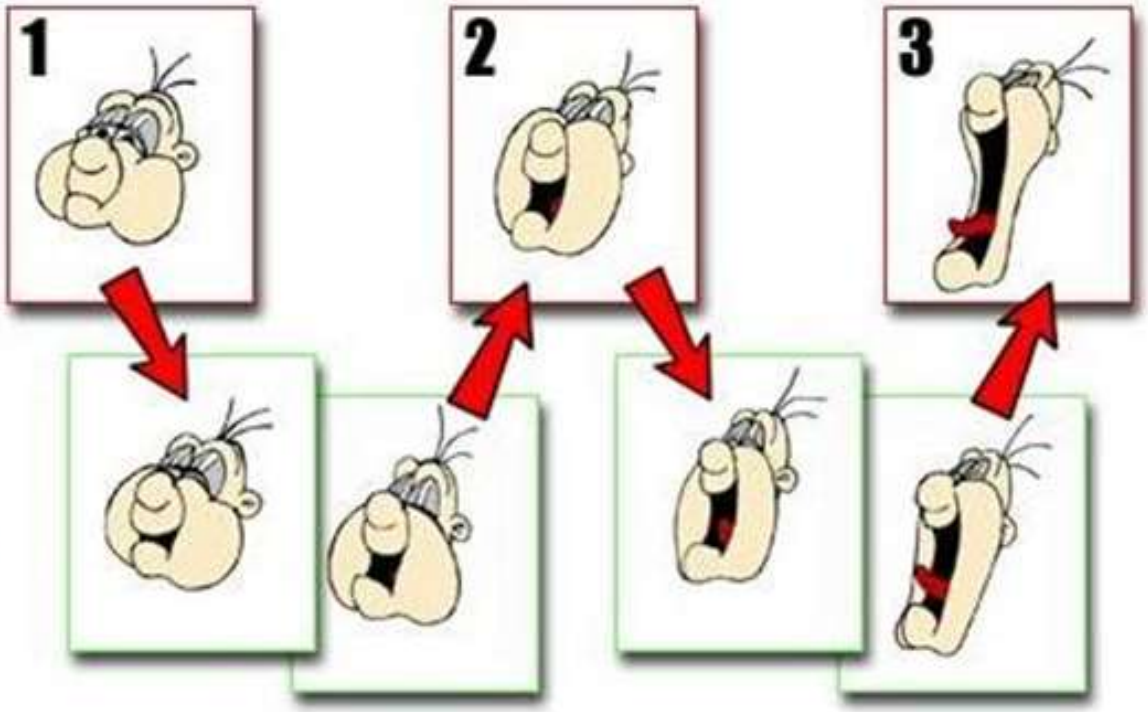
- A keyframe is a detailed drawing of the scene at a certain time in the animation sequence.
- Within each key frame, each object is positioned according to the time for that frame.
- Some key frames are chosen at extreme positions in the action; others are spaced so that the time interval between key frames is not to great.

4. Generation of in-between frames:

- In-between frames are the intermediate frames between the key frames.
- The number of in-between frames is based on the media to be used to display the animation. In common, film needs twenty-four frames per second, and graphic terminals are refreshed on the rate of 30 to 60 frames per second.
- Classically, there are three to five in-between frames for each pair of keyframes.
- E.g. We want to design an animation sequence for 10 seconds. For this we need to have 24x10 frames because film requires 24 frames per second. Out of these 240 frames, we can have 48 keyframes and remaining 192 in-between frames. In this case there are 4 in-between frames between two keyframes.

KeyFrames

In-between frames



• Animation languages

- There are many different languages for describing computer animation and new ones are constantly being developed.
- Animation languages can be categorized as:

1) Linear-list notations

2) General purpose languages

3) Graphical languages

- **1) Linear-list notations:**

- In Linear-list notations, each event in the animation is described by a starting and ending frame number and an action that is to take place.
- The action is specified by a statement with relative parameters.
- E.g. 30,45, C ROTATE “ARM”, 1,60
- The above statement say that - between frames 30 and 45, rotate the object called ARM about axis 1 by 60^{o} , determining the amount of rotation at each frame from table C.

• 2) General purpose languages:

- General purpose languages such as C, C++, Pascal or Lisp can be used to design and control the animation sequences.
- These languages have great potential. They can certainly do everything that Linear-list notations do. But most of these languages require considerable programming expertise.
- ASAS is an example of general purpose language.

➔ It is built on top of LISP.

➔ Its primitives include vectors, colours, polygons, solids and lights.

➔ It also includes a wide range of geometric transformations that operate on objects. These transformations take an object as an argument and return a value that is transformed copy of the object. These transformations include up, down, left, right, zoom-in, zoom-out, forward and backward.

3) Graphical languages:

- There are several specialized animation languages developed called graphical languages.
- These languages provide various animation functions which make it easy to design and control the animation.
- These animation functions include a graphics editor, a keyframe generator, an in-between generator and standard graphics routines.
- Thus, specialized animation language allows us to design and modify object's shape and position. They also define photometric parameters such as light intensities, surface illumination properties. They can set camera parameters such as position, orientation and lens characteristics.
- Following are some graphical languages:
- **1) Keyframe systems 2) Parameterized systems 3) Scripting systems.**

- **1) Keyframe systems:**
- This is a specialized animation language used to generate in-between frame from the user-specified keyframes.
- By controlling the movement of object bodies, keyframe systems generate in-between frames.

- **2) Parameterized systems:**
- These languages specifies the object motion characteristics to be as a part of the object definitions.
- So the designer can have control over object's characteristics such as degree of freedom, motion limitation and allowable shape changes.

- **3) Scripting systems:**
- This language allows object specifications and animation sequences to be defined in a user-input script.
- It is possible to write a script for defining object and its motion.
- Such a scripts can be stored in the library.

Morphing

- Morphing is transformation of object shape from one form to another
- Morphing is common in entertainment industry. Morphing is widely used in movies, animation, games etc.
- Morphing is the process in which the source image is gradually distorted and vanished while producing the target image.

- The earlier images in the sequence are similar to source image and last images are similar to target image.
- Middle image of the sequence is the average of the source image and the target image.

- When an object is described using polygon, we compare the two keyframes for which in-between frames are to be generated.
- The keyframes are compared in terms of number of vertices and number of edges . If they are unequal, they are added or deleted to match the count as a preprocessing steps.
- This is shown in following example:

➔ There are 2 keyframes, K and K+1.

➔ The frame K has one line segment while frame K+1 has two line segments.

➔ As keyframe K+1 has an extra vertex, we add a vertex between vertices 1 and 2 in keyframe K to balance the number of vertices in two frames as a preprocessing step.

➔ Then using linear interpolation, we translate the added vertex in keyframe K into vertex 3' along the straight line path.

➔ So the intermediate position of this 3' gives us the in-between frames.

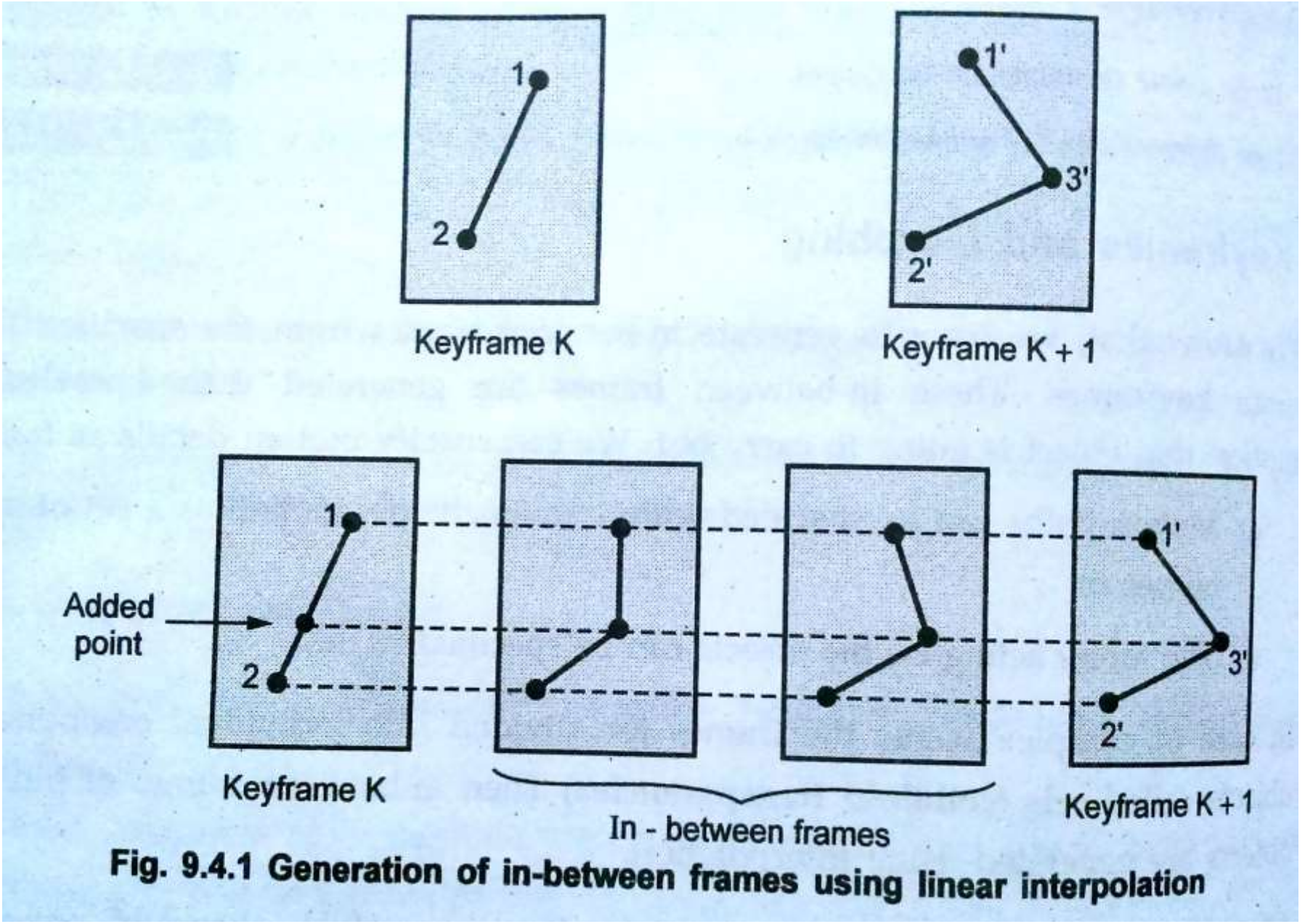
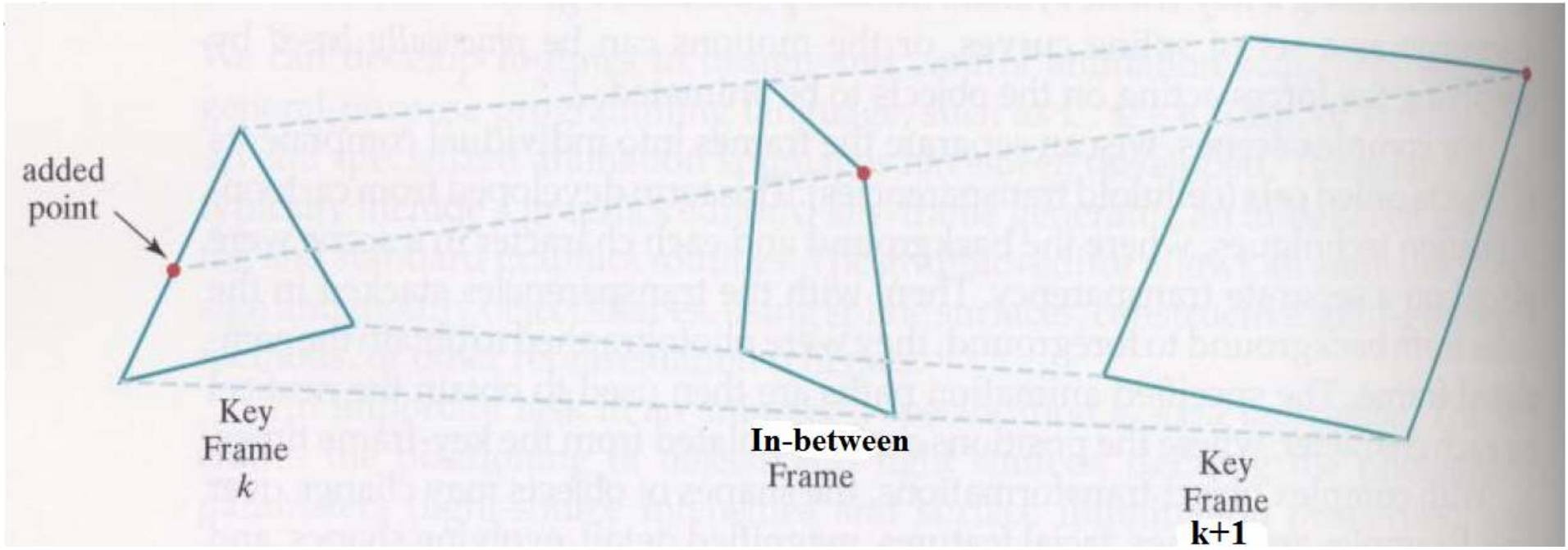


Fig. 9.4.1 Generation of in-between frames using linear interpolation

Following figure shows transformation of triangle into quadrilateral using linear interpolation:



Equalize two keyframes

- Edge equalization
  - The maximum and minimum number of lines to be equalized

$$L_{max} = \max(L_k, L_{k+1})$$

$$L_{min} = \min(L_k, L_{k+1})$$

  - Where L_k and L_{k+1} denote the number of line segments in two successive frames
  - Compute the quantities

$$N_e = L_{max} \bmod L_{min}$$

$$N_s = \text{int}\left(\frac{L_{max}}{L_{min}}\right)$$

  - Preprocessing steps for edge equalization
    - Divide N_e edges if keyframe_{min} into N_s +1 sections
    - Divide the remaining lines of keyframe_{min} into N_s sections

- As an example, if $L_K = 3$ and $L_{K+1} = 11$, then $n_e = 11 \bmod 3 = 2$ and $N_S = \text{int}(11/3) = 3$. Therefore, the preprocessing is accomplished by dividing 2 edges of L_K into 4 sections and dividing the remaining line of L_K into 3 sections. This is illustrated in Fig. 9.4.3.

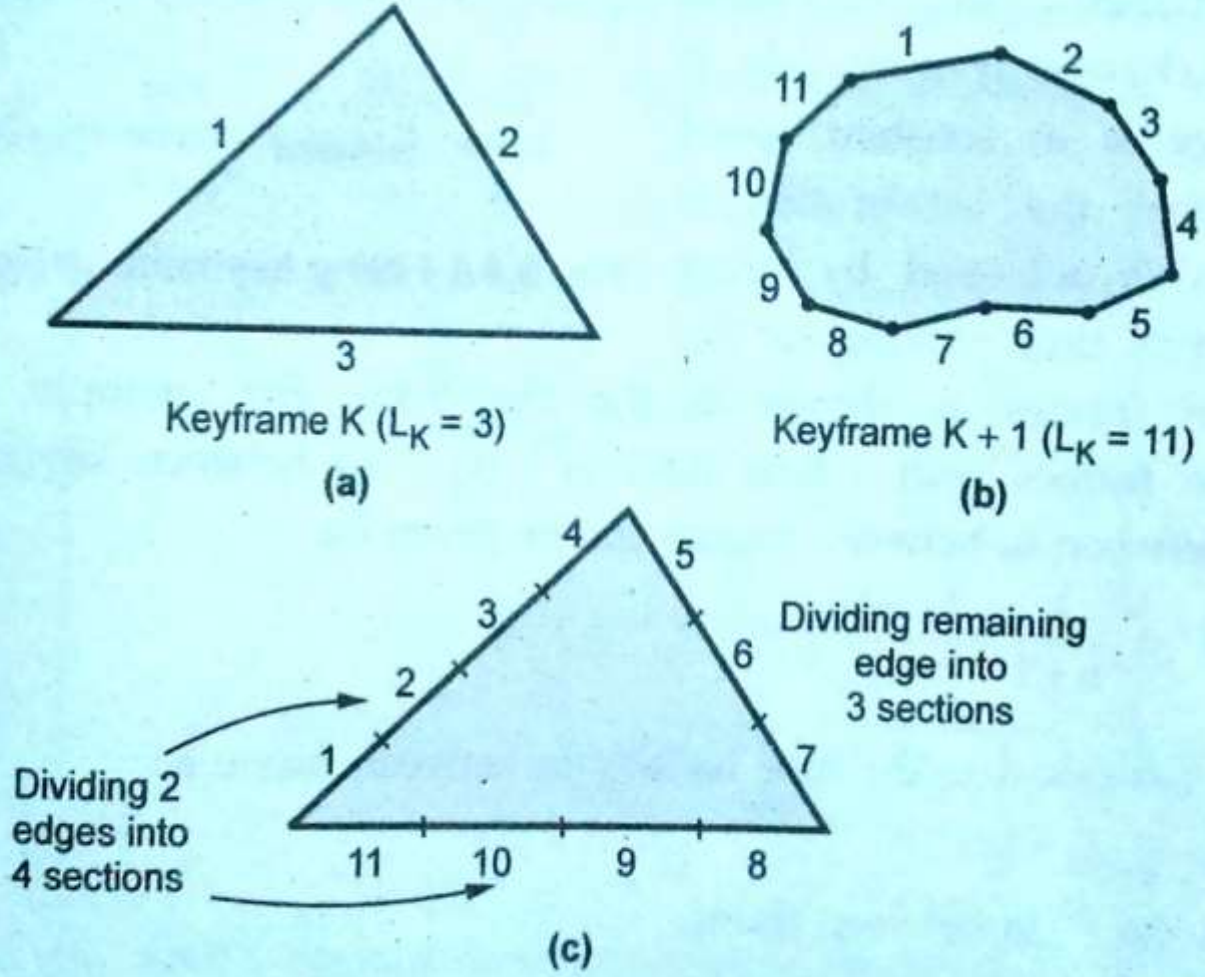


Fig. 9.4.3 General rules for preprocessing

- Vertex equalization
  - The maximum and minimum number of vertices

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

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

  - Where V_k and V_{k+1} denote the number of vertices in two successive frames
  - Compute the quantities

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

$$N_p = \operatorname{int}\left(\frac{V_{\max} - 1}{V_{\min} - 1}\right)$$

  - Preprocessing steps for vertex equalization
    - Add N_p points to N_{ls} line section of $\text{keyframe}_{\min}$
    - Add N_p-1 points to remaining edges of $\text{keyframe}_{\min}$

Tweening

- The animator draws objects and characters either by hand or with a computer.
- Then he positions his creations in key frames, which form an outline of the most important movements.
- Next, the computer uses mathematical algorithms to fill in the "in-between" frames. This process is called **tweening.**
- Key framing and tweening are traditional animation techniques that can be done by hand, but are accomplished much faster with a computer.

Motion specification

- There are following common ways in which the motions of objects can be specified:

- **Direct motion specification**
- **Goal-directed systems**
- **Kinematics and Dynamics**

➔ Direct motion specification:

- It is most straight forward method for defining a motion sequence.
- In this method, the rotation angles and the translation vectors are specified so that geometrical transformations can be applied to the objects in the scene to generate animation sequences.
-

➔ Goal-directed systems:

- In these systems, instead of specifying motion parameters, goal specific instructions are specified.
- For example: We would specify that we want an object to walk or to run to a particular destination.

➔ Kinematics and Dynamics:

- **1) Kinematic descriptions:**
- Motion parameters such as position, velocity and acceleration are specified without reference to the forces that causes the motion to generate animation sequences.
- **Inverse Kinematics:** In this approach, initial and final positions of objects are specified and from that motion parameters are computed by system to generate animation sequences.
- **2) Dynamic descriptions:**
- The forces that produce the velocities and accelerations are specified. Such descriptions of objects are referred as a *physically based modeling.*
- Here, object motions are obtained from the force equations describing physical laws, such as Newton's laws of motion for gravitational and friction forces, Euler or Navier-stokes equations describing fluid flow and Maxwell's equations for electromagnetic forces.