On Intermedia Animation by Image Looping Methods

Frank Appiah

July 29, 2021

On Intermedia Animation By Image Looping Methods.

Frank Appiah [1]

Abstract. On understanding the intermedia animation with image-loop method using list, stack and utility variations like even-indexing, odd-indexing and more. Image-Looping on photo-image, graphics object or graph figure creates a motion picture of still images with delayed seconds in between each frame.

Keywords. Image-Loop, intermedia, animation, even indexing, odd indexing, frame.

1 Introduction

Animation is a way of representing a sequence of intermedia objects like image, infographics and graphs. The sequence of indexes of images created series is of frame animation displays. Animation[1,5] is the graphic art which occurs in time. Whereas a static image (such as a Picasso or a complex graph) may convey complex information through a single picture, animation conveys equivalently complex information through a sequence of images seen in time. It is characteristic of this medium, as opposed to static imagery, that the actual graphical information at any given instant is relatively slight. The source of information for the viewer of animation is implicit in picture change: change in relative position, shape, and dynamics. Therefore, a computer is ideally suited to making animation" possible" through the fluid refinement of these changes. This paper[2] introduces spatial keyframing, a technique for performance-driven character animation. In traditional temporal keyframing, key poses are defined at specific points in time: ie, we define a map from a set of key times to the configuration space of the character and then extend this map to the entire timeline by interpolation. By contrast, in spatial

[1] *Frank Appiah is with Kwame Nkrumah University of Science and Technology,Department of Computer Engineering, Kumasi, Ghana. Since 2012.*

keyframing key poses are defined at specific key positions in a 3D space where the character lives; the mapping from the 3D space to the configuration space is again defined by interpolation. The user controls a character by adjusting the position of a control cursor in the 3D space; the pose of the character is given as a blend of nearby key poses. The user thus can make expressive motion in real time and the resulting motion can be recorded and interpreted as an animation sequence. Although similar ideas are present in previous systems, our system is unique in that the designer can quickly design a new set of keyframes from scratch, and make an animation without motion capture data or special input devices. Our technique is especially useful for imaginary characters other than human figures because we do not rely on motion-capture data. We also introduce several applications of the basic idea and give examples showing the expressiveness of the approach.
[3] introduces digital animation as an arts-based research medium by laying a theoretical foundation for its use and describing how it can become a participatory methodology. The authors link research through digital animation to performance, ethnodrama, film, photography, and visual arts traditions leading to a rationale for using animation as a qualitative research tool. A vignette of an ongoing ethnography contextualizes animation as a process and as a product. In this chapter, the authors argue that digital animation (1) facilitates the use of metaphorical imagery to vividly and emotively capture lived experiences and (2) invites a unique audience into the research discourse
An algorithm animation (AA[3]) visualizes the behavior of an algorithm by producing an abstraction of both the data and the operations of the algorithm. Initially it maps the current state of the algorithm into an image, which then is animated based on the operations between two succeeding states in the algorithm execution. Animating an algorithm allows for better understanding of the inner workings of the algorithm, furthermore it makes apparent its deficiencies and advantages thus allowing for further optimization.
A frame just like a picture frame holds in a container of some decoration. Frame count is the number of frames in a sequence. It is the size of indexes. A transition index has a sequence of indexes of images. Transition can be used to apply position for one or more sequences of intermedia objects. With transition indexing, it is possible to serial animate. This is done in order to create a motion of auto-update image display. Simply, to make a movie of an image-world. Lets keep our hands

clean. Imagine having thousands of images captured by a camera and want to artificially create an animation either in .gif format or .mpeg format. Gif stands for graphics interchange format whiles .mpeg[7, 8] also moving picture experts group. The Graphics Interchange Format[6] is a bitmap image format that was developed by a team at the online services provider CompuServe led by American computer scientist Steve Wilhite and released on 15 June 1987. Animation is the noun for motion pictures in delay time, T expressed as frame per second. A frame per second is the frame rate for inanimate. For a thousand images with t delay in seconds, it will take approximately :

$$T = t \times 1000 \text{ seconds.}$$

2 Image-Loops Method

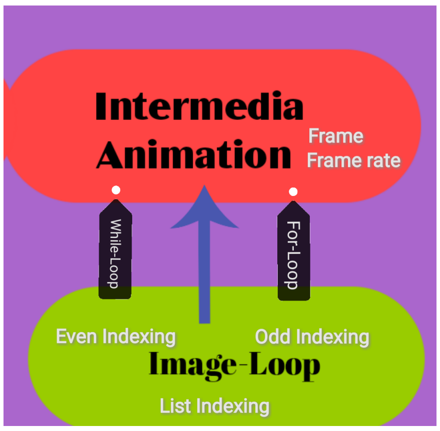
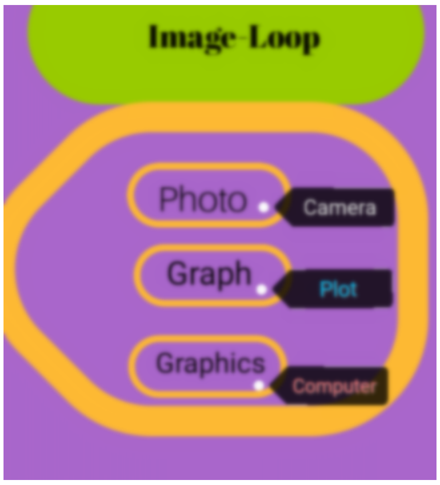


Image-Loop is the cause of an intermedia animation by a controller and effect motion of pictures on display. Image controller is a simple control statement made by a for-loop or while-loop. An organizational view like a stack of pictures will operate like display and pop. This organization will increase the delay time by a deletion time. For thousand of images, it take in seconds approximately:

Figure 1: Details of Image-Loop.

$$T=(t + t1) *1000, \text{ where } t \text{ is delay for display and } t1 \text{ for pop.}$$

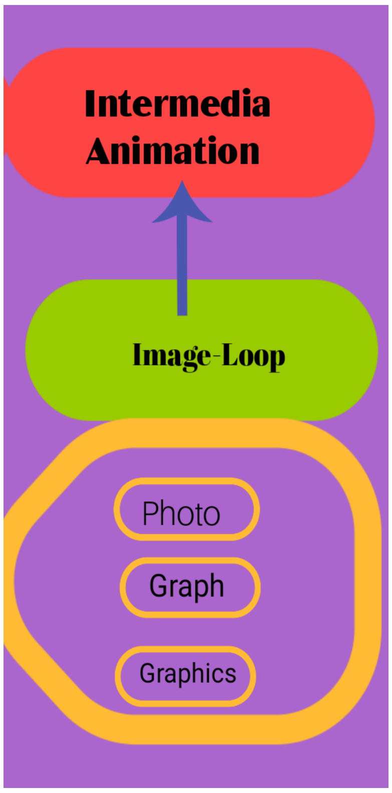
Again an advantage will be memory reduction in file size (fize) . Everytime an image displays, afterwards it gets removed from stack to a poplist. An organizational view like a list of pictures will operate as display and increment indexing. This organization will not increase the delay time because it access on index position. Increment indexing is normally a plus plus - index plus one increasing. Increase Indexing by one can be changed to numeric variants like even indexing, odd indexing or prime indexing. Even indexing will be indexes on even numbers - a plus two indexes. An organizational view of LIFO (last-in first out) will operate like a reverse of pictures in file. These organisations of virtual goods is to simply create a delay enough to cater for rendering time. An exponential delay of rendering time is added to the overall delay.



$T=(t + t1*(e(x) +1)) *1000$, where t is delay for raster display and t1 for accessing file.

Lets look at some pseudocode on the several case scenarios :

Let img[incfm] be container utility for all images.
Let incfm be the index of image.
Let fmcnt be the total desire number of images.
Let display be the show function.
Let frame be the image to display.



1) Stack Image-Loop Pseudocode.

Let stk be the stack utility

Image For-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);
for(incfm;incfm<fmcnt;incfm++)
{
3: stk.push(img[incfm]);
4: frame=stk.pop();
5: display(frame) ;
}
```

Image While-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);
while(incfm<fmcnt)
{
```

```
3: stk.push(img[incfm]);
4: frame=stk.pop();
5: display(frame) ;
6: incfm++;
}
```

2) List Image-Loop Pseudocode.

Let lst be the list utility

Image For-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);

for(incfm;incfm<fmcnt;incfm++)
{
    3: lst.add(img[incfm]);
    4: frame=lst.get(incfm);
    5: display(frame) ;
}
```

Image While-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);
while(incfm<fmcnt)
{
    3: lst.add(img[incfm]);
    4: frame=stk.get(incfm);
    5: display(frame) ;
    6: incfm++;
}
```

3) Even-Indexing List Image-Loop Pseudocode.

Let lst be the list utility

Image For-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);

for(incfm;incfm<fmcnt;incfm+=2)
{
```

```
3: lst.add(img[incfm]);
4: frame=lst.get(lst.lastindex());
5: display(frame) ;
}
```

Image While-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);
while(incfm<fmcnt)
{
    3: lst.add(img[incfm]);
    4: frame=stk.get(lst.lastindex());
    5: display(frame) ;
    6: incfm+=2;
}
```

4) Odd-Indexing List Image-Loop Pseudocode.

Let lst be the list utility
Let odd be a list utility

Image For-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0, odd);

for(incfm;incfm<odd.size();incfm++)
{
    3: lst.add(img[odd[incfm]]);
    4: frame=lst.get(lst.lastindex());
    5: display(frame) ;
}
```

Image While-Loop::

```
1: loadfromFile(img);
2: initialize(incfm=0);
while(incfm<odd.size())
{
    3: lst.add(img[odd[incfm]]);
    4: frame=lst.get(lst.lastindex());
    5: display(frame) ;
    6: incfm++;
}
```

3 Conclusion

Prime indexing Image Looping can be easily implemented by following the odd indexing pseudocode. A list builder, prime will be initialized and build by adding prime numbers like 1, 3, 5, 7, to the maximum size limited by the frame count value. The organizational view of several utility containers were looked at in terms of pseudocode like C++ or Java.

References

1. Baecker, R. M. (1969, May). Picture-driven animation. In Proceedings of the May 14-16, 1969, spring joint computer conference (pp. 273-288).
2. Igarashi, T., Moscovich, T., & Hughes, J. F. (2006). Spatial keyframing for performance-driven animation. In ACM SIGGRAPH 2006 Courses (pp. 17-es).
3. Kerren, A., & Stasko, J. T. (2002). Algorithm animation. In Software Visualization (pp. 1-15). Springer, Berlin, Heidelberg.
4. Loveless, D. J., & Bodle, A. (2014). Framing complexity: Digital animation as participatory research. In Academic knowledge construction and multimodal curriculum development (pp. 327-338). IGI Global.
5. Wells, P. (2013). Understanding animation. Routledge.
6. Eppink, J. (2014). A brief history of the GIF (so far). Journal of visual culture, 13(3), 298-306.
7. Sikora, T. (1997). MPEG digital video-coding standards. IEEE signal processing magazine, 14(5), 82-100.
8. Manjunath, B. S., Salembier, P., & Sikora, T. (Eds.). (2002). Introduction to MPEG-7: multimedia content description interface. John Wiley & Sons.