JUST-IN-TIME PIXELS

Mark Mine and Gary Bishop

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

Abstract

This paper describes Just-In-Time Pixels, a technique for generating computer graphic images which are consistent with the sequential nature of common display devices. Motivation for Just-In-Time Pixels is given in terms of an analysis of the sampling errors which will occur if the temporal characteristics of the raster scanned display are ignored. Included in the paper is a discussion of Just-In-Time Pixels implementation issues. Also included is a discussion of approximations which allow a range of tradeoffs between sampling accuracy and computational cost. The application of Just-In-Time Pixels to real-time computer graphics in a see-through head-mounted display is also discussed.

Keywords: Animation, temporal sampling, head mounted displays

1. Introduction

The sequential nature of the raster scanned display has been largely ignored in the production of computer animated sequences. Individual frames (or at best fields) are computed as if all the lines in an image are presented to the observer simultaneously. Though this is true in animation displayed from movie film, it is inconsistent for conventional CRT based displays which presents the individual pixels of an image sequentially over a period of almost 17 milliseconds. In a typical CRT device, the time of display of each pixel is different and is determined by its X and Y coordinate within the image. Though the period between successive scans is short enough to produce the perception of a continuously illuminated display without flicker, the electron beam in fact only illuminates a tiny region of the display at any one time, and this region stays illuminated only for a fraction of a millisecond.

The presentation of an image that was computed as a point sample in time on a display device that sequentially presents the image pixel by pixel represents an incorrect sampling of the computer generated world. Since the position and orientation of the camera and the position of the objects in the world do not necessarily remain fixed during the period of time required for image scanout, the generated pixel values are inconsistent with the state of the world at the time of their display. This incorrect sampling results in perceived distortion of objects in a scene moving relative to the camera. This distortion can easily be seen in a sequence depicting a vertical line moving across the screen. The line will appear to tilt in the direction of motion because the pixels on scanlines further down the screen do not properly represent the position of the line at the time at which they are displayed.

The goal of Just-In-Time Pixels (JITP) is the generation of pixels that are consistent with the state of the world at the time of their display. This state of the world includes both the position and orientation of the virtual camera and the position of all the objects in the scene. By using the computed trajectory and velocity of the camera and objects, each pixel is rendered to depict the relative positions of the camera and objects at the time at which the individual pixel will be displayed.

The intended application for Just-in-Time-Pixels, at the University of North Carolina in Chapel Hill (UNC), is in maintaining registration between the virtual and the real world in a see-through head-mounted-display system. It is our goal to produce a display system that allows the real world to be augmented with computer generated objects that remain registered in 3-space as the user moves through the workspace. An example of such a system is the ultrasound visualization system described by Bajura, Fuchs, and Ohbuchi [Bajura 92]. Failure to account for the distortion produced by the image scanout delay may result in significant misregistration during user head motions.

2. Just-In-Time Pixels

As stated in the introduction, when using a raster display device, the pixels that make up an image are not displayed all at once but are spread out over time. In a conventional graphics system generating NTSC video, for example, the pixels at the bottom of the screen are displayed almost 17 ms after those at the top. Matters are further aggravated when using NTSC video by the fact that not all 525 lines of an NTSC image are displayed in one raster

scan but are in fact interlaced across two fields. In the first field only the odd lines in an image are displayed, and in the second field only the even. Thus, unless animation is performed on fields (i.e. generating a separate image for each field), the last pixel in an image is displayed more than 33 ms after the first. The problem with this sequential readout of image data, is that it is not reflected in the manner in which the image is computed.

Typically, in conventional computer graphics animation, only a single viewing transform is used in generating the image data for an entire frame. Each frame therefore, represents a point sample in time which is inconsistent with the way in which it is displayed. As a result, as shown in figure 1, the resulting image does not truly reflect the position of objects (relative to the view point of the camera) at the time of display of each pixel.

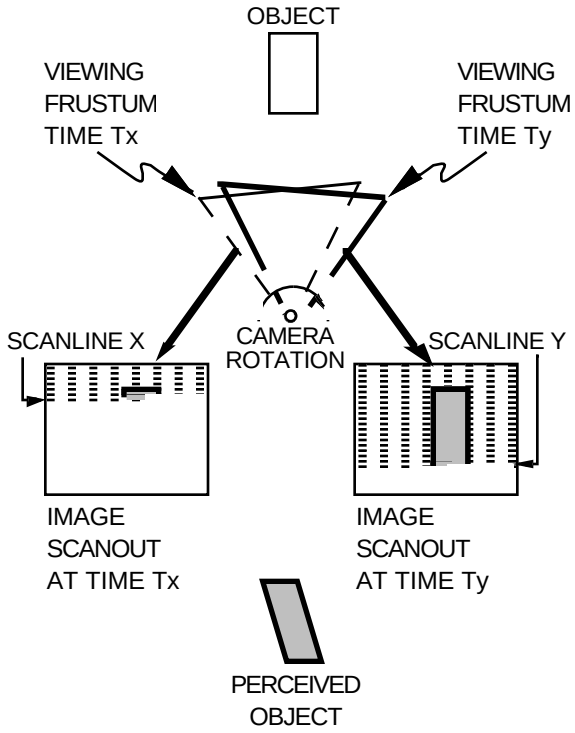

figure 1: Image generation in conventional computer graphics animation

A quick "back of the envelope" calculation can demonstrate the magnitude of the errors that result if this display system delay is ignored. Assuming, for example, a camera rotation of 200 degrees/second (a reasonable value when compared with peak velocities of 370 degrees/second during typical head motion - see [List 83]) we find:

Assume:

1) 200 degrees/sec camera rotation
2) camera generating a 60 degree Field of View (FOV) image
3) NTSC video
   - 60 fields/sec NTSC video
   - ~600 pixels/FOV horizontal resolution

We obtain:

$$200\frac{degrees}{sec}\times\frac{1}{60}\frac{sec}{fields}=3.3\frac{degrees}{field}\text{ camera rotation}$$

Thus in a 60 degree FOV image when using NTSC video:

$$3.3\ degrees\times\frac{1}{60}\frac{FOV}{degrees}\times 600\frac{pixels}{FOV}=33\text{ pixels error}$$

Thus with camera rotation of approximately 200 degrees/second, registration errors of more than 30 pixels (for NTSC video) can occur in one field time. The term registration is being used here to describe the correspondence between the displayed image and the placement of objects in the computer generated world.

Note that even though the above discussion concentrates on camera rotation, the argument is valid for any relative motion between the camera and virtual objects. Thus, even if the camera's view point is unchanging, objects moving relative to the camera will exhibit the same registration errors as above. The amount of error is dependent upon the velocity of the object relative to the camera's view direction. If object motion is combined with rotation the resulting errors are correspondingly worse.

The ideal way to generate an image, therefore, would be to recalculate for each pixel the position and orientation of the camera and the position and orientation of the scene's objects, based upon the time of display of that pixel. The resulting color and intensity generated for the pixel will be consistent with the pixel's time of display. Though objects moving relative to the camera would appear distorted when the frame is shown statically, the distorted JITP objects will actually appear undistorted when viewed on the raster display. As shown in figure 2, each pixel in an ideal Just-In-Time Pixel renderer represents a sample of the virtual world that is consistent with the time of the pixel's display.

In an ideal JITP renderer, therefore, both the viewing matrix and object positions are recomputed for each pixel. Acceptable approximations to Just-In-Time Pixels can be obtained, however, with considerably less computation. Examples include the recomputation of world state only once per scanline and the maintenance of dual display lists.

These methods, and an analysis of the errors they introduce, are discussed below.

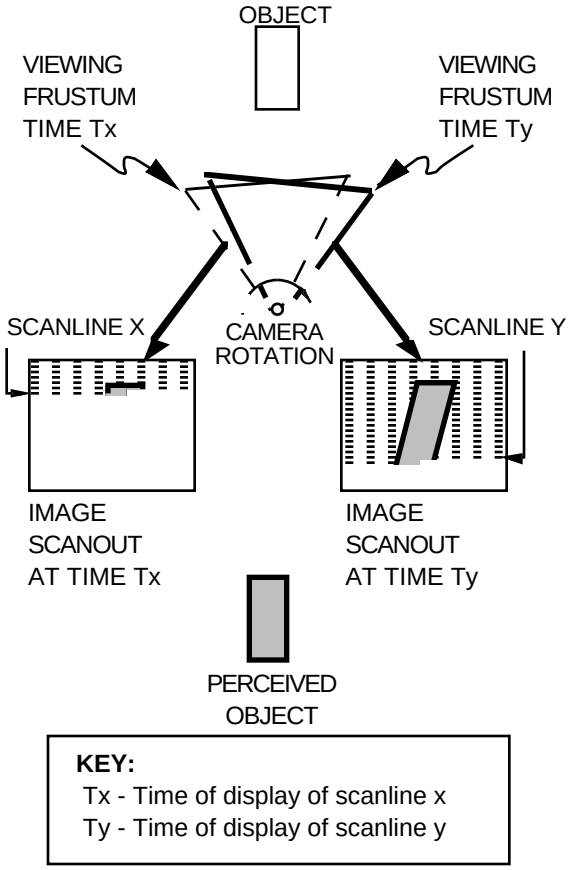

figure 2: Image generation using Just-In-Time Pixels

3. Just-In-Time Pixels Approximations

3.1 Scanline Approximations

Since the time to display a single scanline is small (approximately 65 microseconds for NTSC video) a single viewing transform can be used to compute the pixel data for an entire scanline. Though this represents only an approximation of a true Just-In-Time Pixels renderer (one viewing transform per pixel) it is a significant improvement over the single viewing transform per image used in most conventional graphics systems. This can be shown by performing more "back of the envelope" calculations similar to the ones in section 2.0:

$$200\frac{degrees}{sec}\times\frac{1}{60}\frac{sec}{fields}\times\frac{1}{262.5}\frac{field}{scanlines}$$
$$=0.013\frac{degrees}{scanline}\text{ camera rotation}$$

Thus in a 60 degree FOV image when using NTSC video:

$$0.013\ degrees\times\frac{1}{60}\frac{FOV}{degrees}\times 600\frac{pixels}{FOV}$$
$$=0.13\text{ pixels error}$$

Thus less than a pixel error for NTSC video resolutions.

3.2 Dual Display List Approximation

Another potential simplification reduces the number of transformations required per image while still taking into account the raster scan used in the display of the image. Instead of using a separate viewing transform for each scanline, two display lists are maintained: one representing the position of the objects at the start of display of each field and the second representing the position of the objects at the end of display of that field 16.67 ms later (less the vertical retrace interval). For each field, the data in the two display lists is transformed based upon the predicted view point of the camera at the two times (start of field and end of field). The data for the intermediate scanlines is then generated by interpolating the position of object verticies between the two positions. Thus instead of transforming all the objects for every scanline, each object would be transformed only twice. The resulting scanlines, however, would represent an approximation to Just-In-Time pixels.

For example, if we define:

- T_0 the time of display of the first scanline in an image
- T_1 the time of display of the last scanline in an image
- P_0 position of vertex P at time T_0
- P_1 position of vertex P at time T_1

then we can compute the data for some scanline x:

t_x time of display of scanline x
$P(t_x)$ Position of vertex P at time t_x :
$T_0 \le t_x \le T_1$

where

$$P(t_x)=P_1(t_x-T_0)/(T_1-T_0)$$
$$+P_0(1-(t_x-T_0)/(T_1-T_0))$$

4. Just-In-Time Pixels and Real-Time Computer Graphics

4.1 Application of Just-In-Time Pixels in Real-time Computer Graphics

One of the proposed areas of application of the Just-In-Time-Pixels paradigm at the University of North Carolina, is in the generation of images for see-through head-mounted displays (HMDs). Unlike a conventional HMD, wherein the user is totally

immersed in a computer generated world, a see-through head-mounted display serves to augment the real world by overlaying computer graphic images on top of the real world. Obstetricians, for example, could view ultrasound data overlaid on top of a patients body, "seeing" the ultrasound image of the baby and its position relative to the mother [Bajura 92]. Similarly, architects could envision building modifications in place, getting a better understanding of the interaction of the proposed changes with the existing structure.

Thus it can be seen that the registration of virtual images and real world objects in a see-through HMD is of paramount importance. Lack of proper registration will result in virtual objects being incorrectly positioned relative to the real world (a virtual glass, for example, might be floating several mm above the real surface it's supposed to be resting on). This registration, which is dependent upon many factors such as tracking system accuracy and distortion due to the HMD optics, is complicated by the presence of delays due to image scanout discussed in section 2.0. The use of a single viewing transform to compute the data for an entire image results in errors in the registration of virtual objects and the real world just as it results in errors in the sampling of the computer generated world in computer animation.

Currently at UNC, therefore, implementation is underway of a system that renders images using the Just-In-Time-Pixels paradigm. This system is intended to be used in a see-through HMD to help reduce the registration errors between virtual objects and the real world. In a real-time JITP system, instead of computing pixel values based upon the predicted position and velocity of the virtual camera, each pixel is computed based upon the position and orientation of the user's head at the time of display of that pixel. Generation of a Just-In-Time pixel in real time, therefore, requires knowledge of when a pixel is going to be displayed and where the user is going to be looking at the time. This implies the continuous and parallel execution of the following two central functions:

1) Synchronization of image generation and image scanout
2) Determination of the position and orientation of the user's head at the time of display of each pixel

By synchronizing image generation and image scanout, the JITP renderer can make use of the details of how the pixels in an image are scanned out to determine when a particular pixel is to be displayed. By knowing what scanline the pixel is on, for example, and how fast the scanlines in an image are displayed, the JITP renderer can easily calculate the time of display of that pixel.

Determination of where the user is looking can be accomplished through use of a conventional head tracking system (magnetic or optical for example). Determination of where the user is looking *at the time of display* of a pixel requires the use of a predictive tracking scheme. This is due to the presence of delays between the sampling of the position and orientation of the user's head and the corresponding display of a pixel. Included in the end-to-end delays is the time to collect tracking data, image generation time and the delays due to image scanout.

Estimation of user viewpoint at the time of display of each pixel is accomplished through use of a Kalman filter. Though impossible to predict the true position of the user's head at pixel display time, the Kalman filter produces an estimated view point based upon the user's past head motions. Note that the predictive tracking techniques depend upon a detailed knowledge of when a pixel is to be displayed in order to accurately predict the future head position for that time.

4.2 Enhancements: Beam Racing and Just-In-Time Pixels

In the current implementation, the calculations for each scanline are pushed as late as possible. Ideally data for each scanline is transferred to the frame buffer just before it is read out by the raster scan. This technique, known as beam-racing, was first used in some early flight simulators. By pushing the graphics calculation as late as possible, beam racing allows image generation delays to be combined with display system delays. The result is lower overall end-to-end delay which simplifies the task of predicting the future position and orientation of the user's head. Prediction also benefits from the fact that the delayed computation makes it possible to use the latest available tracking data in the generation of the predicted user view point.

Though a Just-In-Time-Pixels renderer can benefit from the use of beam-racing, the two are not dependent upon one another and are in fact orthogonal concepts. A Just-In-Time-Pixels renderer, for example, can be implemented without beam-racing. In such a system, image data is computed for the predicted time of scanout of a pixel, but computation of the entire image is completed prior to the scanout of the first pixel. This is the type of system that would be useful in the generation of pre-computed animation sequences (see section 2.0 above).

Alternately beam-racing can be implemented without using Just-In-Time Pixels (as was done in the early flight simulators). In such a system, the entire image is computed as though it were a point sample in time (i.e. one viewing transform is used to compute the pixel values for the entire frame) but

the computation of each pixel (or scanline) is completed just prior to its scanout. The advantage (beyond the reduction of end-to-end delays) is that the size of frame buffers can be minimized since data from only a few pixels and not an entire image must be stored.

4.3 Implementation: Distribution of Computation

Incorporation of the Just-In-Time-Pixels paradigm into a real-time computer graphics system can have a considerable impact on the amount of computation required to generate each image. Even if only a scanline approximation to the Just-In-Time-Pixel paradigm (as described in section 3. above) is implemented, the objects in the computer graphic database will have to undergo a significantly greater number of transformations (by factor of 256 when compared to animation on fields) in order to determine the position of each object (relative to the virtual camera) at the time of display of each pixel. This suggests the need for the distribution of computation across multiple processors.

In the current implementation of Just-In-Time Pixels currently at UNC for example, computation is divided among the multiple graphics processors (GP) available in the Pixel-Planes 5 system. Pixel-Planes 5 is a high-performance, scalable multicomputer for 3D graphics designed and built at UNC. GPs are Intel i860 based processors that are used to perform geometric transformations and other calculations in the Pixel-Planes graphics computer.

In the current JITP system, one GP is designated as the Master GP and is primarily responsible for the synchronization of image computation with display scanout and the distribution of tracking data to the remaining GPs. The remaining GPs are designated as slave GPs. The main task of the slave GPs is the computation of the pixel data for the resulting image. To distribute the computation among the slave GPs, data for successive scanlines is divided among the available GPs in the system. Thus in a system with n-slave GPs, each slave GP is responsible for computing the data for every nth scanline in the image.

By dividing the computation for successive scanlines among multiple GPs, each GP is given sufficient time to calculate the pixel data for a scanline. Without this division of labor, the available computation time (65 microseconds for one scanline) would be insufficient to generate an image of reasonable complexity.

5. Conclusions

Just-In-Time Pixels has been presented as a method for generating images that are consistent with their display: the raster scan out of one pixel at a time. The purpose of this is to eliminate the errors in an image which are a result of using a single viewing transform to compute the data for an entire image. Instead, each pixel is generated so that it is consistent with the user's view at the time of display of that pixel. The development of Just-In-Time Pixels is part of the overall program at UNC to reduce and compensate for the end-to-end delays inherent in Head-Mounted Display systems. The goal: stationary virtual objects that appear stationary in the real world. It is hoped that by minimizing the end-to-end delays and compensating for the nature of the display devices in use, the feeling of presence in the virtual world will be augmented and the feeling of "swimming" through the virtual scene eliminated.

References

[Bajura 1992] Bajura, M., Fuchs, H., Ohbuchi, R., "Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery within the Patient." *Computer Graphics (Proceedings of SIGGRAPH' 92),* **26**(2),pp203-210.

[List 1983] List, U. "Nonlinear Prediction of Head Movements for Helmet-Mounted Displays" Technical Paper AFHRL-TP-83-45, Dec. 1983.