EmbeddedSPARK 2010 Round 2 Description

Randall Maas
6604 Silent Creek Ave SE
Snoqualmie, WA
98065

This is a description of my prototype for the EmbeddedSPARK 2010 challenge. I built an interactive entertainment music player, and scratch table for kids. The case is from a small portable stereo, with the Vortex86 board installed inside, and a Logitech Quickcam 9000 mounted on top and directed down toward the tabletop:

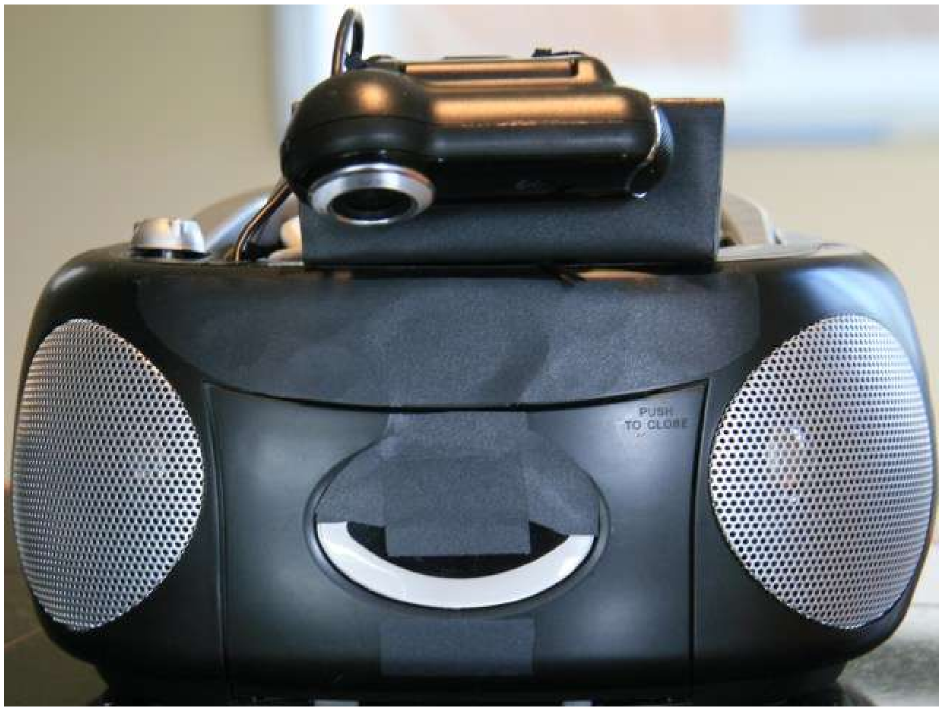

***Figure 1:** Music player*

A kid can play music, like with other music players. But he can also 'scratch' the music backward and forward. In addition, the entire playback controls – e.g. the pausing, scratching the music – are done by a child moving paper shapes on a tabletop.

1 Description of Kids play

The music box is operated by touching, moving and spinning things – behavior that is natural for kids. The child plays with the paper pieces on the table, exploring the rules of play. In addition, – very importantly – he can show other kids the neat tricks he has figured out, as well as the play pieces he has created. This style of interaction and exploration is inspired by the Montessori philosophy.

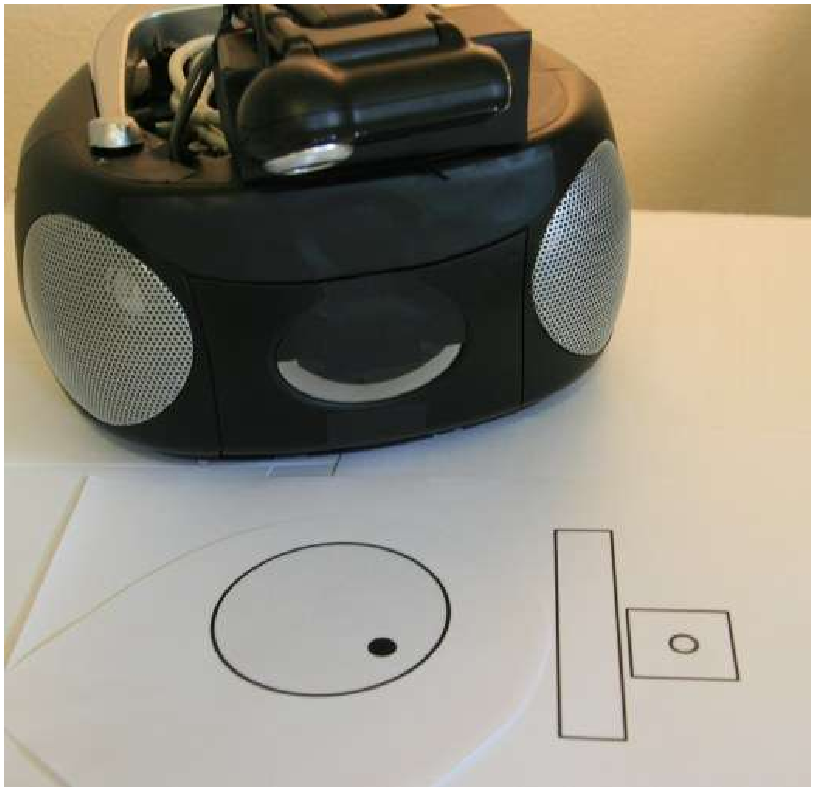

Figure 2: *Player on table*

Note: The term "prototype" is used to distinguish from the ideal productized version.

Controls

Sliders, control buttons, knobs, scratch disks, and so forth on the surface of a table control the music playback. These controls are nested circles and rectangles of various sizes. For example, a solid circle within a rectangle is a button to pause or play music. A circle within (but near the edge of) a circle is a scratch disc, causing the music to skip back a few seconds.

Role of personal craft

The child creates his own controls from paper or cloth, giving him creative influence beyond exploring the sound. He could glue these shapes together, print them from a computer file, or sew them from felt cloth and buttons. (There will be a printable template to get them started.) There is supposed to be something both silly and personal in creating his own controls, intended to build self-esteem and foster sharing (*"I made this!"* and *"Look what I made!"*).

1.1 Principles of experience

To review, these are the principles that the music player is attempting to embody

Principles

1. A *child* is a key part of the overall experience. He may share his experience with other kids.
2. The role of *personal creation* is important to the fun. A kid creates his own pieces (silly and otherwise) to control the music box. Although the kit comes with a template of controls, everyone is encouraged to make their own.
3. The music box should provide a *responsive experience*, encouraging open exploration with appropriate feedback.
4. The primary interaction is mediated by a video camera. The prototype has a power switch and master volume. (The ideal device would not have none.)
5. A child should be able to lift and carry the music box with him.

2 Technical Profile

Physical form

The device is styled like a portable stereo, with a handle and a video camera mounted on the top. Size is important – it is intended to be used and carried by a kid. The camera is pointed at the table top, where it can see the controls. A lamp to illuminate the table can be used. Note: The prototype has a limits on the kind of surface that can be used: the surface must be a solid color tone (e.g. no wood grain texture).

Kinds of controls

Controls. There are three basic kinds of controls that may be made and used on a table's surface:

- Sliders: volume sliders, time position (future development)
- Knobs that control play speed, relative play position, etc
- Buttons – that are connected to actions, e.g. pause, play, on, off, etc.

The software watches the shapes, inferring the controls and their state, and connects these to actions such as playback. Rectangles and circles are easy for software to recognize, determine the size, center, and orientation. This makes it possible to determine changes in their relative positions and orientation.

2.1 Overview of video processing

The system is built to work with a 2D surface. This allows many simplifications:

- Image distortion isn't present or is easy to correct.
- Visual objects don't change scale,
- A visual object doesn't change shape or distorted as it moves across the surface
- Color doesn't matter for the processing, at least for the prototype

Algorithms

The software incorporates traditional algorithms such as JPEG decoding, thresholding, segmentation, trigonometry, and object tracking. The steps in processing are

1. Get frame and decode it into a gray scale image. This consumes most of the processing time.
2. Do background removal. Everything brighter than a threshold is considered background, everything darker is foreground. The threshold is at least 74% of the brightest pixel in the image, and raised so that at least a small percentage of pixels are considered foreground.
3. Separate the image into "blobs" – connected foreground pixels – with a flood fill type of algorithm.
4. Finding corners, outlying points and estimated center of the rectangle (or circle).
5. Determine the shape. To save on processing time, this was simplified to deciding if the shape has the same width and height (which is calls a "circle") or if they differ (which it calls a "rectangle"). With more processing time, e.g. reducing the frame rate, the edge could be tracked in more detail and checked for curved vs straight line characteristics.
6. Map objects to previous object (past)
7. Map object movements and changes to action: pause/resume, scratch music, etc.

As more processing time becomes available – thru software improvements and hardware improvements – other algorithms can be incorporated to improve the quality of device. This might include using smoothing filters and 2D derivatives to edge detection, such as Robert's Cross.

2.2 Video Frame

The video camera's video frames are accessed using modules not part of Windows CE development kit:

- A driver called WebCam.dll is used to get each video frame. This driver was modified to fix some bugs, improve frame rate and work with the model of video camera used.

//cewebcam.codeplex.com/

- A MJPEG decoder, modified from a JPEG image decoder, that decodes each frame into the grayscale Y component (the Cb and Cr components of the image were dropped for performance reasons).

The frame size of 320x240 with a frame rate of 30 frames per second was chosen to allow decoding and processing to happen before the next frame is ready and provide a responsive system.

2.3 Music and Scratch effect

By comparison the audio portions are almost trivial:

- Music File playback is implemented with the DirectShow API.
- The audio effects are implemented with Audio Compression Manager (ACM) and WaveOut API's to decode the sound-effect file and play it, respectively. This approach was chosen to allow more responsive effects by preloading the WAV file.

The "scratch effect" is performed as two parts:

- The DirectShow playback position is set back by 2 seconds,
- The scratch sound effect is played. It seems gratuitous to play such an effect, but it is intended to give a natural response to the action. Without it, the skip back doesn't seem to be responsive to the child's action.

Note: the music files are preloaded onto a CompactFlash in the prototype. A real product may download or accept a CD, scanning it for music.

3 Lessons learned during development

It is common, even desirable, for prototyping projects to have gone down a few dead ends, finding elements that are harder to work with than anticipated. Here are a few I encountered:

The camera:

1. Stock Windows CE doesn't include webcam support. A little research found that the way to access webcam video on CE is to use an add-on called webcam.dll
2. The webcam.dll didn't work with the camera out of the box. It had some small bugs and didn't go above 15fps.
3. The Vortex86 Board Support Package (BSP) has a driver that crashes if the webcam is attached; this affected sound. I created a fake USB driver to attach to the webcam's microphone to prevent the BSP driver from probing the webcam.
4. There wasn't a decoder to convert from MJPEG to a bitmap. I ported one.

5. The MJPEG decoding process took far more time than I expected – almost all of the time I planned to allocate to image analysis.
6. The video frame has extra pixels at the it starts, causing the image to appear to roll around the right edge onto the left side. I don't know if that is something I introduced, a misconfiguration, or other source.[1] I have not resolve this issu yet.

The sound:

1. The DirectSound API was not present. I expected that it would be, interpreting it as part of DirectShow.
2. The Audio Compression Manager was not able to decode MP3's. Again, I expected that this was the (likely) underlying mechanism to audio file decoding in the DirectShow stack. To play MP3's, I used DirectShow API.

4 Conclusion

In conclusion, for the EmbeddedSPARK 2010 challenge I built an interactive entertainment music player, and scratch table for kids. In it is constructed from a small portable stereo, with the Vortex86 board installed inside, and a Logitech Quickcam 9000. It is controlled by paper shapes on a table top.

[1] The MJPEG object appears to be correctly formatted, suggesting it is not a data transfer issue.