D3BOX Task for BCI2000

Jennia Hizver, Gerwin Schalk

July 21, 2003

Contents

Abstract

The primary purpose of this task is the replication of the 2D center-out task (i.e., D2BOX) in a 3D environment. Users can control a cursor towards targets using control in two dimensions. Another purpose of this task is the creation of generic functions that will allow the creation of any task in a simple 3D environment with certain characteristics.

1 Functionality

2 The Environment

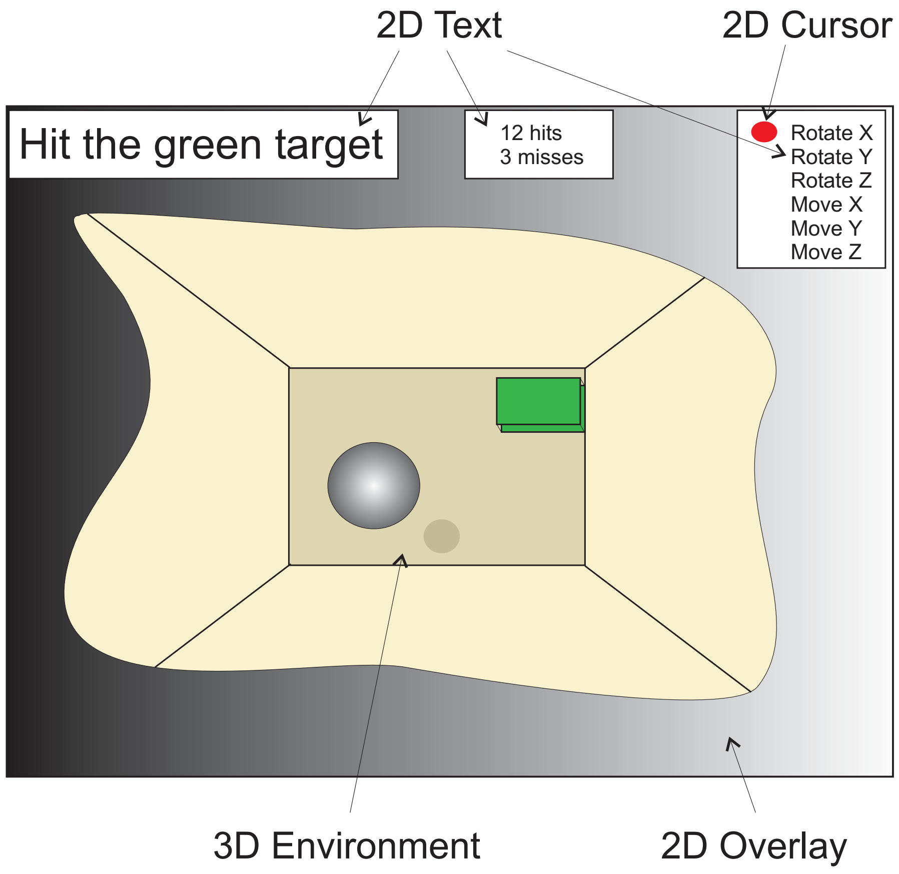

Figure 1: Graphical Environment.

2.1 Rendering

2.2 The 3D Environment

2.2.1 Predefined Graphic Primitives

The 3D environment supports any number of predefined graphic primitives. Each graphic primitive can be assigned a number of different properties. These are:

- Element ID
- Primitive ID
- Primitive-specific properties
- Generic properties
  - Brightness (0-255)
  - Transparency (0-255)
  - Color (RGB)
  - Texture

Supported primitives are spheres (i.e., primitive ID 1), cuboids (i.e., primitive ID 2), and infinite planes (i.e., primitive ID 3). The position of a sphere is defined by midpoint (XYZ) and radius. Textures are mapped using circular mapping using arbitrary alignment. The position of a cuboid is defined by a midpoint (XYZ) and a width, height, and depth (to define the extend in X, Y, and Z, respectively). Texture mapping is defined by the orientation and size of a texture. The only supported mapping type is parallel mapping. Infinite planes are defined by three points in XYZ coordinates. The only supported texture mapping type is parallel mapping with the texture being parallel to the plane.

Once created, there are certain functions that can be applied to any element.

- Change properties
- Turn on/off
- Move midpoint to XYZ
- Test for collision with another element
- Rotate element with rotation defined by
  - Angles in X, Y, Z
  - Reference point in X, Y, Z

2.2.2 Custom Graphic Primitives

In addition to built-in primitives, the system also supports custom graphic primitives. These primitives can be defined by loading their 3D structure from a file and they support the same methods as the predefined primitives except collision detection.

2.2.3 3D Text

The system supports any number of 3D text elements. Each text element is defined by

- Font
- Font size
- Caption
- Brightness (0-255)
- Transparency (0-255)
- Color (RGB)
- Origin (XYZ)
- Direction (XYZ)

The text elements support methods to modify their properties and to turn them on or off.

2.2.4 Camera

The 3D environment supports one camera with the following properties:

- Camera View Point (XYZ)
- Aim Point (XYZ)
- Focal Length
- Camera Orientation (what is "up" for the camera) (angle)

2.2.5 Light Source

The 3D environment supports one omni-directional light source. The features that describe this light source are:

- Position (XYZ)
- Brightness (0-255)
- Color (RGB)

In addition, the brightness of white ambient background lighting can be defined.

2.3 The 2D Overlay

The 3D display can be overlayed by a 2D display (see Figure 1. This overlay supports a picture with alpha channel (i.e., transparency map), 2D text and a 2D cursor.

2.3.1 Overlay Picture

An overlay picture can be defined by loading it from disc. This picture is stretched to match the window size. Optionally, a transparency map can be defined (i.e., in essence, another picture) whose brightness level defines the s transparency of the overlay. This can create an effect as shown in Figure 1. Once defined, an overlay image can be turned on or off.

2.3.2 2D Text

The system supports any number of 2D text elements. Text elements are always displayed on top of the overlay picture. The texts' background is transparent and the text itself does not destroy the overlay picture. For each 2D text element, the following properties can be defined:

- Position (XY)
- Font
- Size
- Color (RGB)

2.3.3 2D Cursor

The system supports one 2D cursor that is always presented as a circle that is filled with a particular color. Specifically, a 2D cursor has the following properties:

- Position (XY)
- Size
- Color (RGB)

State Name	**Bits**	**Description**
TargetCode	8	target ID of the currently active target, or 0 if no target active
ResultCode	8	target ID of the currently selected target, or 0 if no target selected

Table 1: Encoding scheme for this task.

3 Parameters

The following parameters configure this task:

- `WinXpos`

4 States

The time line of stimulus delivery is encoded in state variables as defined in Table 1.

5 Time Line