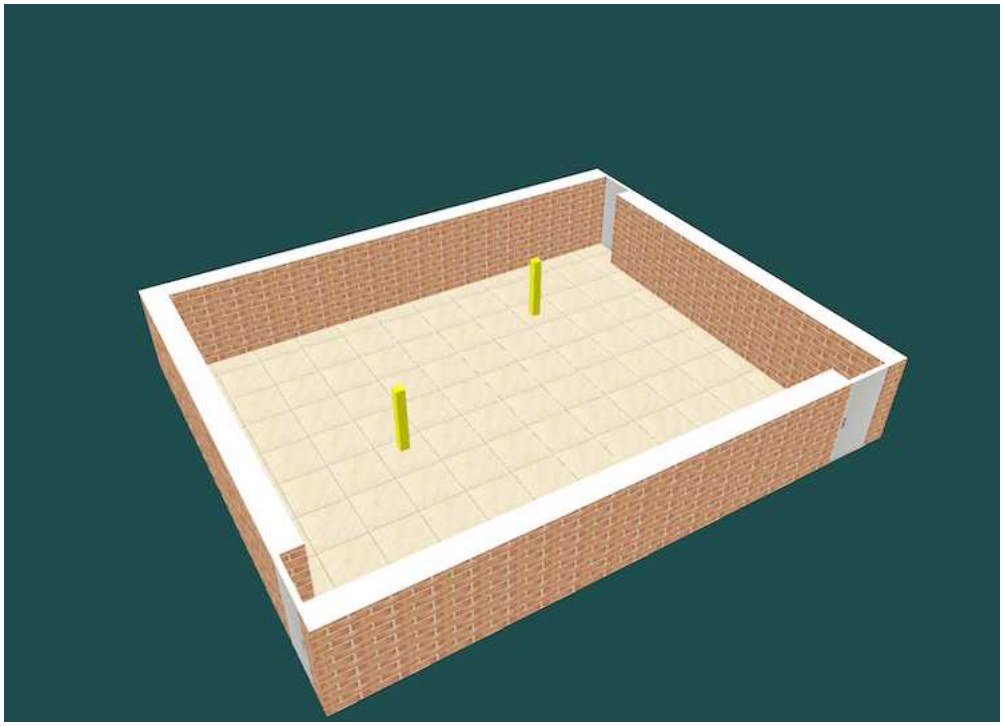
Write and test 3D apps in Desktop Java IDE before building them in C3dapp Studio for Website use

1. Overview

C3dapp Studio is a cloud environment in which you can use Java programming language to write 3D apps for websites and run them in web browsers. You can do the complete 3D app development work in browser without needing to install Java or an IDE on your computer.

However, if you prefer to develop and test 3D apps in Desktop mode before transferring them for website use, you can use the C3dappSDK in a Java IDE on your computer to help writing the Java codes, building and testing the 3D apps locally in your desktop environment before turning the 3D apps into browser mode through C3dapp Studio for website use.

This tutorial guides you through the process to build a 3D app showing a room with three doors. Two people enter the room through the entrance door before splitting up and exiting through the two exit doors separately.



If you haven't already got a Java IDE on your computer, you may download Java SDK from **https://oracle.com**, and download Netbeans IDE from **https://netbeans.org**. You can choose to download and use other Java IDEs instead of Netbeans if you prefer.

C3dappSDK, C3dapp Studio, this document, and the texture image resources used in the tutorial can be downloaded or accessed from **https://oakhousesoftware.com**.

2. Download the C3dappSDK and the texture image files

The free version of C3dappSDK and the image files used in this tutorial need to be downloaded. Open the following download link in a web browser and download the files before saving them in a folder on your computer.

Download link: https://oakhousesoftware.com/downloads/

Files to download:
C3dappSDK-1.0.0
BrickTile.png
DoorBack.png
DoorFront.png
GroundTile.png

3. Prepare to work on the case in Desktop Java IDE

The C3dappSDK comes in a zip file **C3dappSDK-1.0.0.zip**. After unzipping it, you will find the following jar library files in the Libs sub-folder:

dmive-gt-dj-1.0.0.jar
ohsfl-jogamp-1.0.0.jar
ohsfl-utils-1.0.0.jar

In your Desktop Java IDE, create a Java Application project called **RoomWithThreeDoors**.

In Project Properties, add the three jar files from C3dappSDK as Compile-time Libraries. Change Run-time settings so that the Main Class is **`com.ohs.gt.dj.Main`** which is pre-defined in the C3dappSDK.

Create the java package **`com.ohs.gt.client`** in Source Packages. Create a new Java source file called **`GtAppClient.java`** in the package.

Create the java package **`resources`** in Source Packages. Create a new JSON file called **`images.json`** in the package.

4. Prepare the texture image resources for the case

Use your computer's file browser to locate the project's source folder in which you can find two sub-folders: **`com`** and **`resources`**. Get inside the **`resources`** sub-folder, you can find the file **`images.json`**. Copy the following downloaded image files to the **`resources`** sub-folder so that they are alongside the **`images.json`** file:

`BrickTile.png`,
`DoorBack.png`,
`DoorFront.png`, and
`GroundTile.png`

Get back to the **RoomWithThreeDoors** project in your Desktop Java IDE. Open the **`images.json`** file in the **`resources`** package, delete all of the existing lines before adding the following lines in it:

```
[
    "/resources/BrickTile.png",
    "/resources/DoorFront.png",
    "/resources/DoorBack.png",
    "/resources/GroundTile.png"
]
```

5. Write the Java code for the case

Continue to work in the **RoomWithThreeDoors** project in your Desktop Java IDE, open the source file **`GtAppClient.java`** in the package **`com.ohs.gt.client`** for editing.

5.1) Prepare the starting point of the java code

To form the starting point of the java code of the tutorial, delete all the existing coding lines in **`GtAppClient.java`** and replace them by the following lines:

```
package com.ohs.gt.client;

//The external class importing zone
import dmive.core3d.app.AppClient3D;

public class GtAppClient extends AppClient3D {

    //The parameter declaration zone

    @Override
    public void initApp() {
        //The scene initialization zone
    }

    @Override
    public void onAspectRatioChange() {
        //The view configuration zone
    }

    @Override
    public void pickAction(int pickX, int pickY, String keyCmd) {
        //The user interaction zone
    }

    @Override
    public void preUpdate() {
        //The drawing cycle pre-updating zone
    }

    //The internal class definition zone
}
```

Build and run the project. A black background is displayed in the desktop app's window.

5.2) Set background colour and viewing behaviour

In *the external class importing zone*, add the following lines:

```
import dmive.core3d.app.ClientScene3D;
import dmive.core3d.shape.ViewPlatform;
```

In *the parameter declaration zone*, add the following line:

```
    private ClientScene3D cs;
```

In *the scene initialization zone*, add the following lines:

```
        cs = getCs();
        cs.setBgColor(0.12, 0.3, 0.3);
```

In *the view configuration zone*, add the following lines:

```
        ViewPlatform vp = getVp();
        vp.setLookUp(-9.0, -15.0, 14.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0);
        vp.setNearZ(0.1);
        vp.setFarZ(200.0);
        vp.setViewDisMin(0.2);
        vp.setViewDisMax(150.0);
        vp.setvZmin(-2.0);
        vp.setViewingBehavior(1);
        vp.setTouchZoomMul(50.0);
```

Build and run the project. A colour background is displayed in the desktop app's window.

(Continue on next page)

5.3) Build walls for the room

In *the external class importing zone*, add the following line:

```
import dmive.core3d.shape.TexHex;
```

In *the internal class definition zone*, create the **Wall** class by adding the following lines:

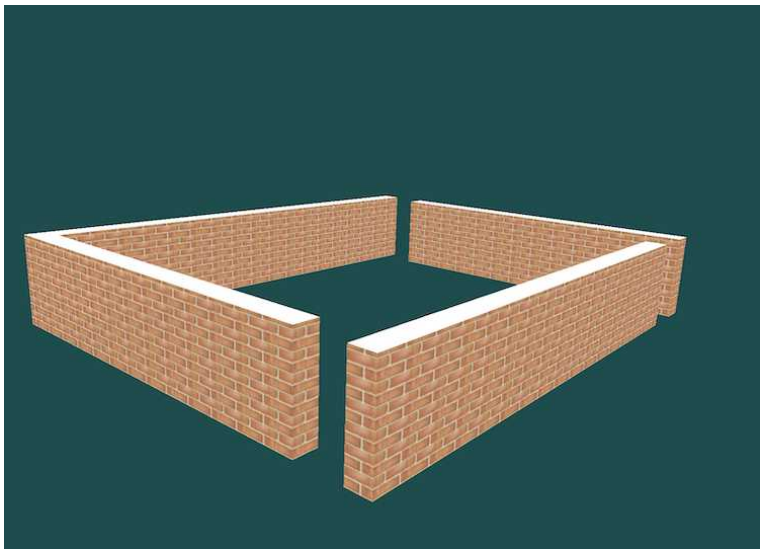
```
    private class Wall extends TexHex {

        private Wall(double x0, double x1, double y0, double y1) {
            super(x0, x1, y0, y1, 0.0, 2.0);
            setWestTex(0, 0.64, 0.32, 0.0, 0.0);
            setEastTex(0, 0.64, 0.32, 0.0, 0.0);
            setSouthTex(0, 0.64, 0.32, 0.0, 0.0);
            setNorthTex(0, 0.64, 0.32, 0.0, 0.0);
            buildShape(cs);
        }
    }
```

In *the scene initialization zone*, add the following lines **after existing lines**:

```
        cs.addShape(new Wall(-6.0, 4.5, -5.0, -4.5));
        cs.addShape(new Wall(5.5, 6.0, -5.0, 3.5));
        cs.addShape(new Wall(-6.0, 6.0, 4.5, 5.0));
        cs.addShape(new Wall(-6.0, -5.5, -3.5, 4.5));
```

Build and run the project. the surrounding walls of the room are displayed. Use mouse or touch points to manipulate the view. Zooming is achieved by mouse wheel rotation or changing the distance between two touch points on a touch screen.



5.4) Fit doors to the room

In *the internal class definition zone*, create the **Door** class by adding the following lines:

```
    private class Door extends TexHex {

        private Door(double rot, double x0, double y0) {
            TexHex doorBody = new TexHex(0.0, 1.0, -0.1, 0.0, 0.0, 2.0);
            doorBody.setSouthTex(1, 1, 1);
            doorBody.setNorthTex(2, 1, 1);
            doorBody.buildShape(cs);
            addChild(doorBody);
            setRotation(rot, 0.0, 0.0, 1.0);
            setTranslation(x0, y0, 0.0);
        }

        //Set door animation
    }
```

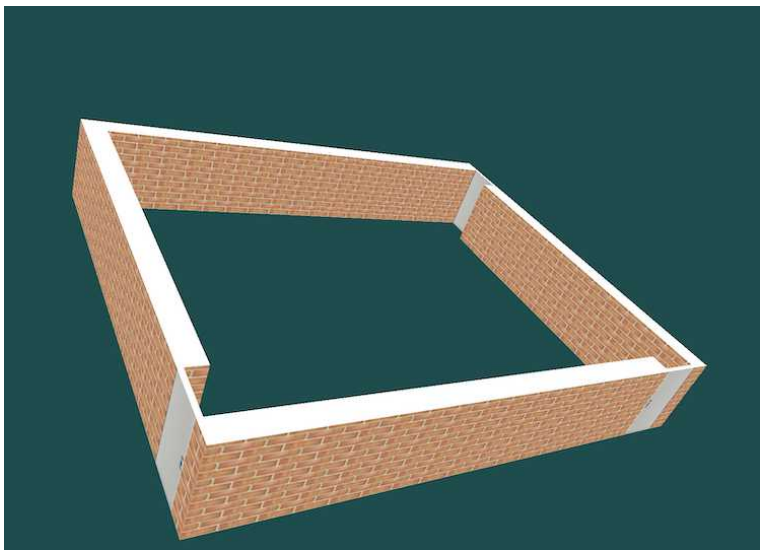
In *the parameter declaration zone* near the top of the file, add the following line:

```
    private Door[] doors;
```

In *the scene initialization zone*, add the following lines **after existing lines**:

```
        doors = new Door[3];
        doors[0] = new Door(-90.0, -5.9, -3.5);
        doors[1] = new Door(180.0, 5.5, -5.0);
        doors[2] = new Door(-90.0, 6.0, 4.5);
        cs.addShape(doors[0]);
        cs.addShape(doors[1]);
        cs.addShape(doors[2]);
```

Build and run the project. Three doors have been added to close the gaps between the walls of the room. The doors won't open at the moment as door animation codes have not yet been added.



5.5) Add people to the scene

In *the external class importing zone*, add the following line:

```
import dmive.core.utils.DVUtils;
```

In *the internal class definition zone*, create the **Person** class by adding the following lines:

```
    private class Person extends TexHex {

        private final int pId;
        private double rpx;
        private double rpy;

        private Person(int pId) {
            super(-0.1, 0.1, -0.1, 0.1, 0.0, 1.5);
            setWestCol(0.9, 0.9, 0.0, 1.0);
            setEastCol(0.8, 0.8, 0.0, 1.0);
            setSouthCol(0.7, 0.7, 0.0, 1.0);
            setNorthCol(0.6, 0.6, 0.0, 1.0);
            setHighCol(1.0, 1.0, 0.0, 1.0);
            buildShape(cs);
            this.pId = pId;
            setPosition(-1.0);
        }

        private void setPosition(double alpha1) {
            if (alpha1 < -0.9) {
                rpx = -4.0 + 8.0 * DVUtils.getRandomInt(1001) / 1000.0;
                rpy = -3.0 + 6.0 * DVUtils.getRandomInt(1001) / 1000.0;
                setTranslation(rpx, rpy, 0.0);
            } else {
                //Set animated position
            }
        }

        //Set person animation
    }
```

In *the parameter declaration zone* near the top of the file, add the following lines:

```
    private Person person0;
    private Person person1;
```

In *the scene initialization zone*, add the following lines **after existing lines**:

```
        person0 = new Person(0);
        person1 = new Person(1);
        cs.addShape(person0);
        cs.addShape(person1);
```

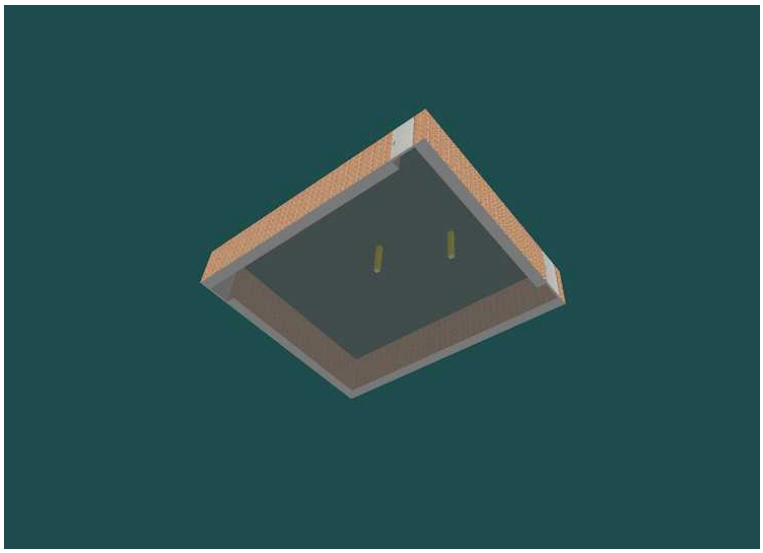
Build and run the project. It shows two people in the room, each taking a random position.

5.6) Add floor for the room

In *the scene initialization zone*, add the following lines **after existing lines**:

```
        TexHex floor = new TexHex(-6.0, 6.0, -5.0, 5.0, -0.002, 0.0);
        floor.setHighTex(3, 1.0, 1.0, 0.0, 0.0);
        floor.setLowCol(0.3, 0.3, 0.3, 0.75);
        floor.buildShape(cs);
        cs.addShape(floor);
```

Build and run the project. It shows the floor in the room. Check the translucent effect by changing the view to look from underneath.



5.7) Open and close doors

Inside the internal class **Door**, add the following lines under the line `//Set door animation`:

```
        private void preUpdate() {
            if (isAnimationOn()) {
                double openFrac = 0.0;
                double alpha = (double) getAnimElapsedTime()
                        / (double) getAnimDuration();
                if (isAnimStarting()) {
                } else if (isAnimStopping()) {
                } else if (!isAnimStopped()) {
                    openFrac = alpha < 0.2 ? alpha / 0.2
                            : (alpha < 0.8 ? 1.0 : (1.0 - alpha) / 0.2);
                }
                getChild(0).setRotation(85.0 * openFrac, 0.0, 0.0, 1.0);
            }
        }
```

In *the parameter declaration zone* near the top of the file, add the following line:

```
    private int animStep = -1;
```

In *the user interaction zone*, add the following lines:

```
        animStep = (animStep + 1) % 3;
        doors[animStep].startAnimation("", 1000);
```

In *the drawing cycle pre-updating zone*, add the following lines:

```
        doors[0].preUpdate();
        doors[1].preUpdate();
        doors[2].preUpdate();
```

Build and run the project. Click with mouse or tap on the display to check the opening and closing of each door.

5.8) People entering and exiting the room

Inside the internal class **Person**, add the following lines under the line `//Set animated position`:

```
                double px;
                double py;
                if (alpha1 < 0.0) {
                    px = -7.0;
                    py = -4.0;
                } else if (alpha1 < 0.3) {
                    px = -7.0 + 2.0 * alpha1 / 0.3;
                    py = -4.0;
                } else if (alpha1 < 0.8) {
                    px = -5.0 + (rpx + 5.0) * (alpha1 - 0.3) / 0.5;
                    py = -4.0 + (rpy + 4.0) * (alpha1 - 0.3) / 0.5;
                } else if (alpha1 < 1.0) {
                    px = rpx;
                    py = rpy;
                } else if (alpha1 < 1.6) {
                    px = rpx + (5.0 - rpx) * (alpha1 - 1.0) / 0.6;
                    py = rpy + (-4.0 + 8.0 * pId - rpy) * (alpha1 - 1.0) / 0.6;
                } else {
                    px = 5.0 + pId * 2.5 * (alpha1 - 1.6) / 0.4;
                    py = 4.0 + (1 - pId) * (-8.0 - 2.5 * (alpha1 - 1.6) / 0.4);
                }
                setTranslation(px, py, 0.0);
```

Inside the internal class **Person**, add the following lines under the line `//Set person animation`:

```
        private void preUpdate() {
            if (isAnimationOn()) {
                if (animStep == 0 || animStep == pId + 1) {
                    double alpha = (double) getAnimElapsedTime()
                            / (double) getAnimDuration();
                    if (isAnimStarting()) {
                        if (animStep == 0) {
                            alpha = 0.0;
                            setActive(true);
                            setPosition(-1.0);
                        }
                    } else if (isAnimStopping()) {
                        alpha = 1.0;
                        setActive(animStep == 0);
                    } else if (!isAnimStopped()) {
                    }
                    setPosition(alpha + (animStep == 0 ? -0.2 * pId : 1.0));
                }
            }
        }
```

In *the scene initialization zone*, add the following lines **after existing lines**:

```
        person0.setActive(false);
        person1.setActive(false);
```

In *the user interaction zone*, add the following lines **after existing lines**:

```
        person0.startAnimation("", 1000);
        person1.startAnimation("", 1000);
```

In *the drawing cycle pre-updating zone*, add the following lines **after existing lines**:

```
        person0.preUpdate();
        person1.preUpdate();
```

Build and run the project. Click with mouse or tap on the display to check the animation effect of the two people entering and exiting the room.

The coding in **`GtAppClient.java`** for the C3dapp of **RoomWithThreeDoors** is now complete.

6. Build the app in C3dapp Studio and run it in Web Browser

After the above steps, when the C3dapp of **RoomWithThreeDoors** is working as expected in Desktop mode with the Java IDE, it's time to build it in C3dapp Studio for website use.

Enter the C3dapp Studio: https://oakhousesoftware.com/c3dapp-studio/

Click on tab 'Case Editor', then click on '**Create a new case**'. Change the Case title to '**RoomWithThreeDoors**'.

Click on '**Add an image**' to open the Image Bundle Organizer. Upload the four image files in the following order before clicking on '**Close and Show Code**' to return to the Case Editor main panel.

image file name	BrickTile.png	DoorFront.png	DoorBack.png	GroundTile.png
pixel size	32 x 16	64 x 128	64 x 128	128 x 128
				
image id	0	1	2	3

In the Desktop Java IDE, open the source file **`GtAppClient.java`** in the package **`com.ohs.gt.client`** of the **RoomWithThreeDoors** project. Copy all lines of the source file into clipboard.

Get back to the Case Editor in C3dapp Studio, delete all of the existing lines from within the code editing box before pasting the clipboard contents into the code editing box.

Make sure that the Case title is **RoomWithThreeDoors**. Click on '**Build C3dapp for the case**' to build and run the case. If the case building is successful, a case with title '**RoomWithThreeDoors**' will appear in the Built Cases panel showing a room with three doors. Click with mouse or tap on the display to check the animation effect of the two people entering and exiting the room.

Click on '**Save and download case**' under case title '**RoomWithThreeDoors**' to save the case on your local computer. The case file with name '**RoomWithThreeDoors.zip**' that has been downloaded can be saved in your chosen folder for later use.

7. Next steps

To find out the ways to use the C3dapps on your website, you may visit **C3dapp Studio** and click on the 'Next Steps' tab.

To learn more about writing C3dapps, you may visit the Tutorials section of the C3dapp Studio.

To see how the C3dapp case of RoomWithThreeDoors can be created step-by-step in C3dapp Studio without needing a local Java IDE, you may take a look at the following article:

C3dapp Studio – a cloud environment for
writing 3D apps in Java and running them on Websites

which is available from **the download page**.

A pre-prepared C3dapp case zip file **RoomWithThreeDoors.zip** containing the completed Java source code and the image resources for this tutorial is also available from the download page.