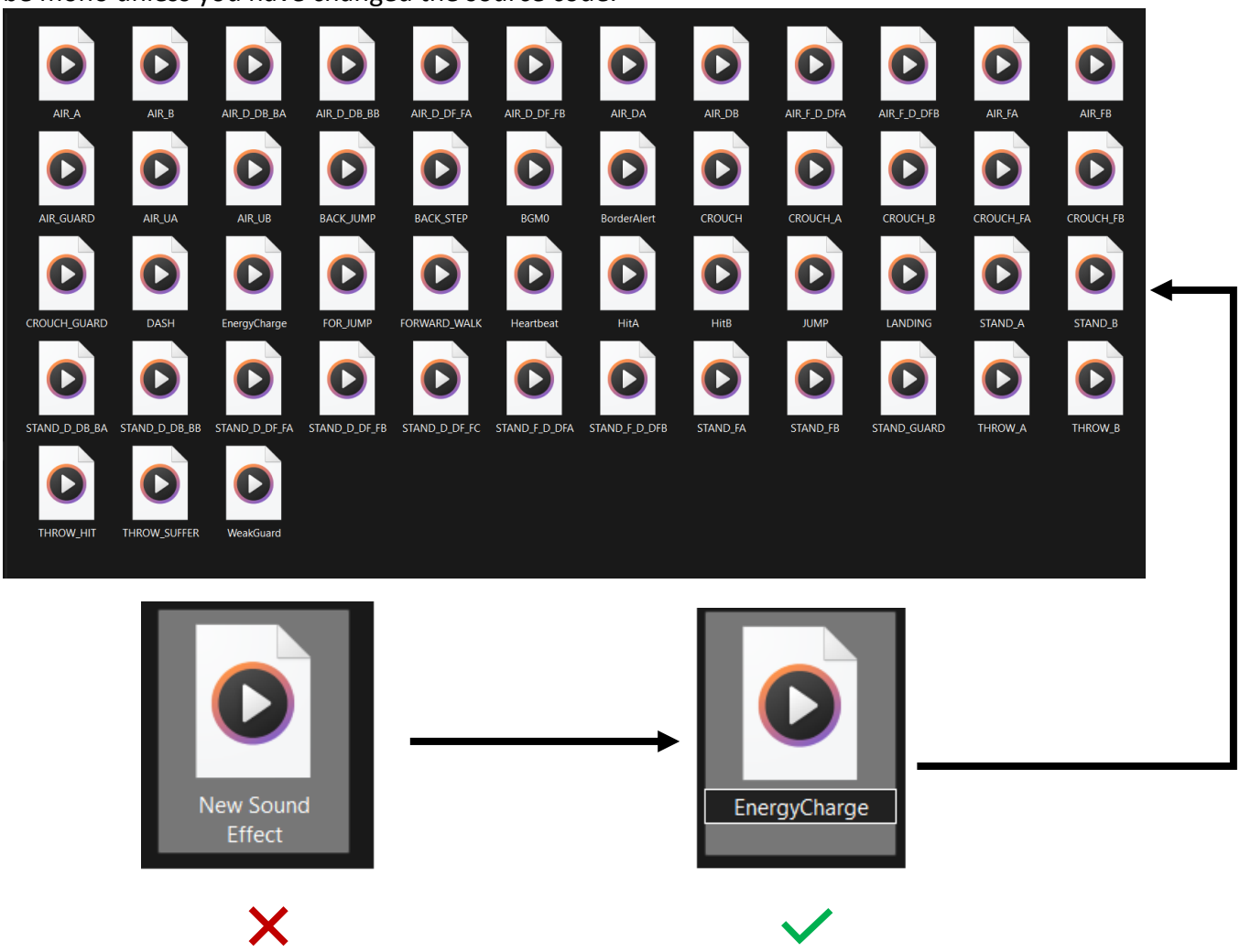
Instructions on Sample Sound Design

Only Changing the Sound Effects:

For the sound design track of the DareFightingICE competition, the simplest sound design you can create is to create or find sound effects for each action of the character and overwrite the previous sound effect with your own sound effect.

The sound effects can be found in “DareFightingICE/data/sounds” folder. While changing the sound effects make sure not to change the name of the sound effect as the names are extremely important. All the sound effects must be in .wav format and the sound effects should be mono unless you have changed the source code.



Sample Sound Design:

The sample sound design uses openAL from the Lightweight Java Game Library (lwjgl), and there is a total of 51 sound effects. You can access the documentation on the openAL from this link.

There are three important elements in our sample design: Listener, Audio-Sources, and Audio-Buffers.

1. Listener, as it is clear by the name, listens to the audio generated by the game. It is responsible for providing the output of audio to the game players. The position of Listener in DareFightingICE is in the middle of the game screen since the camera does not move from its original position. The Listener's velocity is defined by its speed and direction, which is used to create a Doppler effect. Two vectors represent the Listener's orientation: "at" and "up," meaning the forward direction, and the latter represents the upward direction, respectively.
2. Audio-Sources are in-game objects or in-game events that produce sound. In our sample sound design, the position of these Audio-Sources from the Listener determines from which direction the players will hear the sound effect. These Audio-Sources also have velocity and orientation. A Doppler effect is created with the help of the velocity of an Audio-Source of interest and the velocity of the Listener.
3. Audio-Buffers contains all the sound effects that our sample sound design has.

To understand how you can use these three elements to play and modify sound, you can watch the tutorials on openAL 3D audio on this link.

```
public void play2(AudioSource source, AudioBuffer buffer, int x, int y, boolean loop){
    for (int i = 0; i < soundRenderers.size(); i++){
        int sourceId = source.getSourceIds()[i];
        int bufferId = buffer.getBuffers()[i];
        soundRenderers.get(i).play(sourceId, bufferId, x, y, loop);
    }
}
```

Function for playing a sound effect in DareFightingICE/src/manager/SoundManager.java

The above code shows the function to play a sound effect. The *soundRenderers* contains two renderers: one sound renderer for the speakers and the other sound renderer for AIs, and the *bufferId* represents the index of an Audio-Buffer. The parameters *x* and *y* store the horizontal and the vertical locations of where to play the sound effect on the stage (map), respectively, and the *loop* confirms whether to play the sound effect in a loop or not.

```
public void play(int sourceId, int bufferId, int x, int y, boolean loop) {
    set();
    AL10.alSourcei(sourceId, AL10.AL_BUFFER, bufferId);
    alSourcei(sourceId, AL_BUFFER, bufferId);
    alSource3f(sourceId, AL_POSITION, x, 0, 4);
    alSourcei(sourceId, AL_LOOPING, loop ? 1 : 0);
    AL10.alSourcePlay(sourceId);
    int error = alGetError();
}
```

Function to play the sound effect in src/render.audio/SoundRenderer

This piece of code is used to play the sound effects, and it is called by the “play2” function in “SoundManager.java”. The logic of this function is simple since we are only giving the horizontal position (x) to provide the 3D feel. A way to improve this function would be to fine use of the vertical position (y) and to try and to utilize the velocity and the direction of the sound effect by using the attributes “AL_VELOCITY” and “AL_DIRECTION” in the function “alSource3f()”.

The main logic for the sample sound design is in “DareFightingICE/src/fighting/Character.java”. The logic to select which sound effect to play is in a function called "runAction." The function is called every time an action is taken, and it is a simple way to play the right sound effect for the right action by naming the sound effects after the actions themselves.

```
// Playing sound effects based on the actions.
if (FlagSetting.enableWindow && !FlagSetting.muteFlag && !this.isSimulateProcess) {
    if (Arrays.asList("JUMP", "FOR_JUMP", "BACK_JUMP", "THROW_A", "THROW_B", "THROW_HIT", "THROW_SUFFER", "STAND_A", "STAND_B", "CROUCH_A", "CROUCH_B", "AIR_A", "AIR_B", "AIR_DA", "AIR_DB",
            "STAND_FA", "STAND_FB", "CROUCH_FA", "CROUCH_FB", "AIR_FA", "AIR_FB", "AIR_UA", "AIR_UB", "STAND_F_D_DFA", "STAND_F_D_DFB", "STAND_D_DB_BA", "STAND_D_DB_BB", "AIR_F_D_DFA",
            "AIR_F_D_DFB", "AIR_D_DB_BA", "AIR_D_DB_BB").contains(Name)) {
        Name = Name + ".wav";

        SoundManager.getInstance().play2(sourceid,SoundManager.getInstance().getSoundEffect().get(Name),this.x,this.y,false);
        SoundManager.getInstance().play2(sourceDefault, SoundManager.getInstance().getSoundBuffers().get(Name), this.x, this.y, false);

    } else if (Arrays.asList("CROUCH").contains(Name)) {
        Name = Name + ".wav";
        if (!TempName.equals(Name)) {
            SoundManager.getInstance().play2(sourceid,SoundManager.getInstance().getSoundEffect().get(Name),this.x,this.y,false);
            SoundManager.getInstance().play2(sourceDefault, SoundManager.getInstance().getSoundBuffers().get(Name), this.x, this.y, false);
            TempName = Name;
        }
    } else if (Arrays.asList("FORWARD_WALK", "DASH", "BACK_STEP").contains(Name)) {
        Name = Name + ".wav";
        if (!TempName2.equals(Name)) {
            SoundManager.getInstance().play2(sourceid3,SoundManager.getInstance().getSoundEffect().get(Name),this.x,this.y,true);
            SoundManager.getInstance().play2(sourceWalking, SoundManager.getInstance().getSoundBuffers().get(Name), this.x, this.y, true);
            TempName2 = Name;
        }
    } else if (Arrays.asList("STAND_D_DF_FA", "STAND_D_DF_FB", "AIR_D_DF_FA", "AIR_D_DF_FB", "STAND_D_DF_FC").contains(Name)) {
        for(int a = 0 ; a<this.isProjectileLive.length ; a++) {
            if(!this.isProjectileLive[a]) {
                this.isProjectileLive[a] = true;
                sY[a] = this.y;
                sX[a] = this.x;
                Name = Name + ".wav";
                SoundManager.getInstance().play2(sourceProjectTiles[a], SoundManager.getInstance().getSoundBuffers().get(Name), this.x, this.y, true);
                System.out.println(a);
                break;
            }
        }
```

In the above code we filter through all the actions to find the action being taken by the character and then play the sound effect related to that function. It is a brute force way of picking the right sound effect to play. More information about the sample sound design’s logic and is inside the “Character.java” file.