Instructions and Tips

In this document, we are going to give you more details about how our Sample Sound Design works and how you can modify certain things to make it better. Keep in mind that these tips are just general suggestions to give you an idea of what you can do while making your own sound design. To make sure you understand everything, please read the documentation on Sample Sound Design before reading this. All the code is from DareFightingICE/src/manager/ Character.java

```
} 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;
        }
```

The above code is to play the sound of the projectiles when they are called. In the Sample Sound Design, an array of three Audio-Sources is given for each player (P1, P2). These are hardcoded and if somehow there is a 4th projectile, from the same player, visible on the screen it will not emit any sound. Even though it is quite impossible for four projectiles from the same player to be on the screen at the same time, it can still be improved with a better design. In such a design, the number of Audio-Sources will not be fixed.

```
for(int a  = 0 ; a < this.projectileAttack.length ; a++) {
    if(this.projectileAttack[a] != null) {
        if(this.isProjectileLive[a]) {
            if (this.projectileAttack[a].updateProjectileAttack() && !this.ProjectileHit[a]) {

                this.sX[a] = this.sX[a] + (this.projectileAttack[a].getSpeedX());
                this.sY[a] = this.sX[a] + (this.projectileAttack[a].getSpeedY());
                SoundManager.getInstance().SourcePos(sourceProjectTiles[a], this.sX[a], this.sY[a]);
            } else {
                SoundManager.getInstance().stop(sourceProjectTiles[a]);
                this.isProjectileLive[a] = false;
                this.projectileAttack[a] = null;
                this.ProjectileHit[a] = false;
            }

        }
    }
}
```

This particular code is changing the positions of the above-mentioned Audio-Sources according to the position of the projectiles. In our sample sound design, this is accomplished by making a temporary copy of the projectile attack and using its parameters to update the Audio-Source. There are also checks in place to stop the sound effect when the projectile hits the enemy or when it runs out of time and disappears. The drawback of this approach is that the sound comes after the visuals. Another improvement is thus to try and sync the visual and sound perfectly.

```
if(FlagSetting.limitHpFlag) {
    if(this.hp < 50) {
        if(!SoundManager.getInstance().isPlaying(sourceHeartBeat)) {
            if(this.playerNumber)SoundManager.getInstance().play2(sourceHeartBeat,SoundManager.getInstance().getSoundBuffers().get("Heartbeat.wav"),0,0,false);
            else SoundManager.getInstance().play2(sourceHeartBeat,SoundManager.getInstance().getSoundBuffers().get("Heartbeat.wav"),GameSetting.STAGE_WIDTH,0,false);
        }
    }
}
```

The function of this piece of code is to play the “HeartBeat” sound effect when the player’s HP goes below 50. There is room for improvement on this as well. You can change the frequency of the sound effect as the HP gets lower and lower giving the Visually Impaired players the information that they are getting closer to losing.

```
if (this.energy > this.preEnergy + 50) {
    this.preEnergy = this.energy;
    if (this.playerNumber) {
          SoundManager.getInstance().play2(sourceid5,SoundManager.getInstance().getSoundEffect().get("EnergyCharge.wav"),0,0,false);
        SoundManager.getInstance().play2(sourceEnergyChange, SoundManager.getInstance().getSoundBuffers().get("EnergyCharge.wav"), 0, 0, false);
    } else {
          SoundManager.getInstance().play2(sourceid5, SoundManager.getInstance().getSoundEffect().get("EnergyCharge.wav"), GameSetting.STAGE_WIDTH, 0, false);
        SoundManager.getInstance().play2(sourceEnergyChange, SoundManager.getInstance().getSoundBuffers().get("EnergyCharge.wav"), GameSetting.STAGE_WIDTH, 0, false);
    }
```

This code is to play the “Energy Change” sound effect once the player’s energy goes over 50 from the previous time it was played. The Improvement that can be done here is similar to the HeartBeat sound effect. You can increase the frequency when the energy gets to 100, 200, and 300, to give the Visually Impaired players the information that their character's energy is at a higher level.