Отчет по практической работе №13

Студента группы №81

Лазо Константина Олеговича

**Используя практическую работу №11-12 дополнить экран игры исправным таймер и функцией паузы при котором будет прекращаться движение монстров, будет недоступно передвижение героя по клавишам и замирание таймера. Подробное описание работы таймера и функционала паузы смотреть в WSR\_2\_MODULE.docx.**

**Результат:**

Я создал отдельные классы интерфейса игрока (HUD) и для паузы (Pause). Функция таймер работает в классе интерфейса, основная логика паузы находится в главном классе игры, Game, она останавливает всё, что входит в requestAnimationFrame(). Кроме того, сделал стили для паузы.

****

Функция из главного класса игры:

*/\*\**

*\* Функция для requestAnimationFrame*

*\* Здесь же логика паузы*

*\*/*

    frame(*time*)

    {

*// проверка на паузу*

        if(!**this**.isPaused)

        {

**this**.pause.cancel()

*// механизм переключения сцен: если сцена неактивна, то берём следующую*

            if(**this**.currentScene.status != Scene.WORKING)

            {

**this**.currentScene = **this**.changeScene(**this**.currentScene.status);

**this**.currentScene.init();

            }

**this**.currentScene.render(*time*);

        }

        else

        {

**this**.pause.init()

        }

        requestAnimationFrame(*time* => **this**.frame(*time*))

    }

Класс паузы:

*/\*\**

*\* Класс паузы*

*\*/*

export class Pause

{

    constructor(*game*)

    {

**this**.game = *game*;

**this**.screen = **this**.game.scenes.gameProcess.scene;

**this**.div = document.querySelector('#pause');

        const self = **this**;

*// пауза по нажатию на esc*

        window.onkeydown = function(*e*) {

            const key = *e*.keyCode

            switch(key) {

              case 27: self.game.isPaused = !self.game.isPaused; break

            }

        }

    }

    init()

    {

*// стоп таймера*

**this**.game.hud.paused = 0;

**this**.game.hud.pauseTimer();

*// стоп регенерации*

**this**.game.player.intervalTimer = 0;

*// эффект блюра при паузе*

**this**.screen.style.filter = 'blur(5px)';

**this**.div.hidden = false;

    }

    cancel()

    {

**this**.div.hidden = true;

**this**.screen.style.filter = '';

**this**.game.hud.paused = 1;

**this**.game.hud.pauseTimer();

    }

}

Секундомер:

setTimer()

    {

        const self = **this**;

        var timerArray = **this**.timer.innerHTML.split(':');

        function startTimer()

        {

            if(!self.running){

                self.startTime = new Date().getTime();

                self.tInterval = setInterval(getShowTime, 1);

                self.paused = 0;

                self.running = 1;

            }

        }

        function getShowTime()

        {

            let updatedTime = new Date().getTime();

            if (self.savedTime){

                self.difference = (updatedTime - self.startTime) + self.savedTime;

            } else {

                self.difference =  updatedTime - self.startTime;

            }

            var minutes = Math.floor((self.difference % (1000 \* 60 \* 60)) / (1000 \* 60));

            var seconds = Math.floor((self.difference % (1000 \* 60)) / 1000);

            minutes = (minutes < 10) ? "0" + minutes : minutes;

            seconds = (seconds < 10) ? "0" + seconds : seconds;

            self.timer.innerHTML = timerArray[0] + ': ' + minutes + ':' + seconds

        }

        startTimer()

    }

    pauseTimer()

    {

        if (!**this**.difference)

        {

*// если таймер не стартовал*

        }

        else if (!**this**.paused)

        {

            clearInterval(**this**.tInterval);

**this**.savedTime = **this**.difference;

**this**.paused = 1;

**this**.running = 0;

        } else {

**this**.setTimer();

        }

    }