class Main extends eui.UILayer {

protected createChildren(): void {

super.createChildren();

GameLogic.getInstance().GameStage = this.stage;

GameLogic.getInstance().main = this;

egret.lifecycle.addLifecycleListener((context) => {

// custom lifecycle plugin

})

egret.lifecycle.onPause = () => {

egret.ticker.pause();

}

egret.lifecycle.onResume = () => {

egret.ticker.resume();

}

let assetAdapter = new AssetAdapter();

egret.registerImplementation("eui.IAssetAdapter", assetAdapter);

egret.registerImplementation("eui.IThemeAdapter", new ThemeAdapter());

this.runGame().catch(e => {

console.log(e);

})

}

private async runGame() {

await this.loadResource()

this.createGameScene();

await platform.login();

const userInfo = await platform.getUserInfo();

}

private async loadResource() {

try {

const loadingView = new LoadingUI();

this.stage.addChild(loadingView);

await RES.loadConfig("resource/default.res.json", "resource/");

await this.loadTheme();

await RES.loadGroup("preload", 0, loadingView);

this.stage.removeChild(loadingView);

}

catch (e) {

console.error(e);

}

}

private loadTheme() {

return new Promise((resolve, reject) => {

// load skin theme configuration file, you can manually modify the file. And replace the default skin.

let theme = new eui.Theme("resource/default.thm.json", this.stage);

theme.addEventListener(eui.UIEvent.COMPLETE, () => {

resolve();

}, this);

})

}

protected createGameScene(): void {

GameLogic.getInstance().openStart();

}

}

class LoadingUI extends egret.Sprite implements RES.PromiseTaskReporter {

public constructor() {

super();

this.createView();

}

private textField: egret.TextField;

private createView(): void {

this.textField = new egret.TextField();

this.addChild(this.textField);

this.textField.y = 300;

this.textField.width = 480;

this.textField.height = 100;

this.textField.textAlign = "center";

}

public onProgress(current: number, total: number): void {

this.textField.text = `Loading...${current}/${total}`;

}

}

class AssetAdapter implements eui.IAssetAdapter {

public getAsset(source: string, compFunc:Function, thisObject: any): void {

function onGetRes(data: any): void {

compFunc.call(thisObject, data, source);

}

if (RES.hasRes(source)) {

let data = RES.getRes(source);

if (data) {

onGetRes(data);

}

else {

RES.getResAsync(source, onGetRes, this);

}

}

else {

RES.getResByUrl(source, onGetRes, this, RES.ResourceItem.TYPE\_IMAGE);

}

}

}

declare interface Platform {

getUserInfo(): Promise<any>;

login(): Promise<any>

}

class DebugPlatform implements Platform {

async getUserInfo() {

return { nickName: "username" }

}

async login() {

}

}

if (!window.platform) {

window.platform = new DebugPlatform();

}

declare let platform: Platform;

declare interface Window {

platform: Platform

}

class ThemeAdapter implements eui.IThemeAdapter {

public getTheme(url: string, onSuccess: Function, onError: Function, thisObject: any): void {

function onResGet(e: string): void {

onSuccess.call(thisObject, e);

}

function onResError(e: RES.ResourceEvent): void {

if (e.resItem.url == url) {

RES.removeEventListener(RES.ResourceEvent.ITEM\_LOAD\_ERROR, onResError, null);

onError.call(thisObject);

}

}

if (typeof generateEUI !== 'undefined') {

egret.callLater(() => {

onSuccess.call(thisObject, generateEUI);

}, this);

}

else {

RES.addEventListener(RES.ResourceEvent.ITEM\_LOAD\_ERROR, onResError, null);

RES.getResByUrl(url, onResGet, this, RES.ResourceItem.TYPE\_TEXT);

}

}

}

declare var generateEUI: { paths: string[], skins: any }

class GameCommand extends egret.EventDispatcher {

public constructor() {

super();

}

private static \_instance: GameCommand;

public static getInstance(): GameCommand {

if (this.\_instance == null) {

this.\_instance = new GameCommand();

}

return this.\_instance;

}

public sendData(b: boolean = false) {

if (b) {

DataBase.debt = Math.floor(DataBase.debt \* 1.15);

DataBase.deposit = Math.floor(DataBase.deposit \* 1.04);

}

let msg = this.getData();

GameLogic.getInstance().gameui.initData(msg);

}

public sendMarket(evt: boolean) {

DataBase.events = [];

DataBase.marketGoods = [];

let msg: msgGoodsBuyRsp = this.getMarket(evt);

DataBase.marketGoods = msg.goods;

GameLogic.getInstance().gameui.initMarket(msg);

}

public sendStore() {

let msg = new msgGoodsStoreRsp();

msg.goods = DataBase.storeGoods;

GameLogic.getInstance().gameui.initStore(msg);

}

public sendEvent() {

this.dealOtherEvent();

let arr = DataBase.events;

for (let i: number = 0; i < arr.length; i++) {

GameLogic.getInstance().gameui.eventAppear(arr[i]);

}

DataBase.events = [];

}

public sendError(i: number) {

GameLogic.getInstance().gameui.errorRsp(i);

}

public sendOver(t:number) {

DataBase.gameState = 0;

if(t == 0){//时间到

DataBase.debt = Math.floor(DataBase.debt \* 1.15);

DataBase.deposit = Math.floor(DataBase.deposit \* 1.04);

DataBase.money = DataBase.money + DataBase.deposit - DataBase.debt + this.getStorePrice();

DataBase.debt = 0;

DataBase.deposit = 0;

}

else if(t == 1){//体力为0

}

this.sendData();

GameLogic.getInstance().gameui.over();

}

private getStorePrice():number{

let p:number = 0;

for(let i:number=0;i<DataBase.storeGoods.length;i++){

let good = DataBase.storeGoods[i];

p += good.dwPrice \* good.dwNum;

}

return p;

}

public getData(): msgLifeDataRsp {

let msg = new msgLifeDataRsp();

msg.dwMoney = DataBase.money;

msg.dwDebt = DataBase.debt;

msg.dwDeposit = DataBase.deposit;

msg.dwPow = DataBase.pow;

msg.dwTimes = DataBase.times;

msg.dwMaxStoreNum = DataBase.maxStoreNum;

msg.dwFame = DataBase.fame;

return msg;

}

private bases: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];

public getMarket(evt: boolean): msgGoodsBuyRsp {

let msg = new msgGoodsBuyRsp();

msg.goods = [];

let len = 4 + Math.floor(Math.random() \* 6);

let arr = this.bases.slice();

let lll = DataBase.gamePackage < 2 ? arr.length : arr.length - 1;

let goodIds = [];

for (let i: number = 0; i < len; i++) {

let i = Math.floor(Math.random() \* lll);

goodIds.push(arr[i]);

arr.splice(i, 1);

}

goodIds.sort(this.sortfun);

for (let i: number = 0; i < goodIds.length; i++) {

let good = new varGoods();

let id = goodIds[i];

let o = GameLogic.getInstance().goods[id];

if (o == null) {

continue;

}

good.dwID = id;

good.strName = o['name'];

good.dwPrice = this.getPrice(o, evt);

good.dwNum = 0;

msg.goods.push(good);

}

return msg;

}

private dealOtherEvent() {

let b = Math.random() < 0.2;

if (b) {

let a = Math.floor(Math.random() \* 4) + 1;

let b = Math.random() < 0.5 ? 1 : 2;

let c = Math.floor(Math.random() \* 3) + 1;

this.addEvent(a, b, c);

}

}

private addEvent(a, b, c) {

let id = a \* 100 + b \* 10 + c;

let o = GameLogic.getInstance().goods["evt" + id];

if (o == null) {

return;

}

let isadd = b == 2;

let value = o['value'];

if (a < 5) {

switch (a) {

case 1://money

if (value <= 1) {

value = Math.floor(DataBase.money \* value);

}

else {

value = Math.floor(Math.random() \* value / 5);

}

DataBase.money = DataBase.money + (isadd ? value : -value);

DataBase.money = DataBase.money <= 0 ? 0 : DataBase.money;

break;

case 2://deposit

if (value <= 1) {

value = Math.floor(DataBase.money \* value);

}

else {

value = Math.floor(Math.random() \* value / 5);

}

DataBase.deposit = DataBase.deposit + (isadd ? value : -value);

DataBase.deposit = DataBase.deposit <= 0 ? 0 : DataBase.deposit;

break;

case 3://pow

DataBase.pow = DataBase.pow + (isadd ? value : -value);

DataBase.pow = DataBase.pow <= 0 ? 0 : DataBase.pow;

break;

case 4://fame

DataBase.fame = DataBase.fame + (isadd ? value : -value);

DataBase.fame = DataBase.fame <= 0 ? 0 : DataBase.fame;

break;

}

}

if(typeof(o) == "string"){//其他事件

DataBase.events.push(StringUtil.getSwfLangStr(o));

}

else{//商品事件

DataBase.events.push(StringUtil.getSwfLangStrVar(o['str'], [value]));

}

}

private diss1: number[] = [0.2, 5, 10];

private getRandom1(): number {

let r = Math.random();

if (r < 0.1) {

let i = Math.floor(Math.random() \* 3);

return this.diss1[i];

}

else {

return 1;

}

}

private diss2: number[] = [0.1, 0.2, 5, 10];

private getRandom2(): number {

let i = Math.floor(Math.random() \* 4);

return this.diss2[i];

}

private getPrice(o: Object, evt: boolean): number {

let n = o['price'];

let r1 = this.getRandom1();

let v = Math.floor(n \* r1);

let b2 = Math.random() < 0.5;

let v2 = Math.floor(v \* Math.random() \* 0.2);

v = b2 ? v + v2 : v - v2;

if (evt) {

let b3 = Math.random() < 0.1;

if (b3) {

let r3 = this.getRandom2();

v = Math.floor(v \* r3);

let r4 = Math.floor(Math.random() \* 3) + 1;

let evt = 'evt' + (r3 < 1 ? 0 : 1) + r4;

DataBase.events.push(o[evt]);

}

}

return v;

}

private sortfun(a: number, b: number): number {

return a < b ? -1 : 1;

}

private saveAchieve(){

if(DataBase.money > DataBase.achives[0]){

DataBase.achives[0] = DataBase.money;

}

if(DataBase.deposit > DataBase.achives[1]){

DataBase.achives[1] = DataBase.deposit;

}

if(DataBase.debt > DataBase.achives[2]){

DataBase.achives[2] = DataBase.debt;

}

if(DataBase.pow < DataBase.achives[3]){

DataBase.achives[3] = DataBase.pow;

}

if(DataBase.fame < DataBase.achives[4]){

DataBase.achives[4] = DataBase.fame;

}

if(DataBase.fame > DataBase.achives[5]){

DataBase.achives[5] = DataBase.fame;

}

}

private getPriceInMarket(id: number): number {

let arr = DataBase.marketGoods;

for (let i: number = 0; i < arr.length; i++) {

let good = arr[i];

if (id == good.dwID) {

return good.dwPrice;

}

}

return null;

}

public selectPackage(i: number) {

DataBase.gamePackage = i;

}

public startGame() {

let o = GameLogic.getInstance().data["config" + DataBase.gamePackage];

DataBase.times = 1;

DataBase.money = o['money'];//new Int64(o['money'], 0);

DataBase.debt = o['debt'];

DataBase.deposit = 0;//new Int64(0, 0);

DataBase.pow = o['pow'];

DataBase.maxStoreNum = 100;

DataBase.fame = o['fame'];

DataBase.marketGoods = [];

DataBase.storeGoods = [];

DataBase.events = [];

DataBase.achives = [0,0,0,0,0];

DataBase.gameState = 1;

this.sendData();

this.sendMarket(false);

}

public passOneDay() {

if (DataBase.gameState == 0) {

return;

}

DataBase.times++;

if (DataBase.times >= 40) {

this.sendOver(0);

return;

}

this.sendMarket(true);

this.sendEvent();

this.sendData(true);

if (DataBase.pow <= 0) {

this.sendOver(1);

return;

}

this.saveAchieve();

}

public buyGoods(id: number, num: number) {

if (num == 0) {

this.sendError(ERROR.BUY\_ZERO);

return;

}

if(id == 9 && DataBase.gamePackage != 3){

this.sendError(ERROR.NEED\_LICIENCE);

return;

}

let arr = DataBase.marketGoods;

for (let i: number = 0; i < arr.length; i++) {

let good = arr[i];

if (good.dwID == id) {

let n = good.dwPrice \* num;

if (n > DataBase.money) {

this.sendError(ERROR.MONEY\_NOT\_ENOUGH);

return;

}

else {

let arr1 = DataBase.storeGoods;

let total = 0;

let index;

for (let j: number = 0; j < arr1.length; j++) {

let good1 = arr1[j];

if (good1.dwID == id) {

index = j;

}

total += good1.dwNum;

}

if (total + num > DataBase.maxStoreNum) {//柜子不够

this.sendError(ERROR.STORE\_NOT\_ENOUGH);

return;

}

else {

DataBase.money -= n;

let g = arr1[index];

if (g == null) {

g = new varGoods();

g.dwID = id;

g.dwPrice = good.dwPrice;

g.dwNum = num;

g.strName = good.strName;

arr1.push(g);

}

else {

let nn = g.dwNum + num;

let p = Math.floor((g.dwPrice \* g.dwNum + good.dwPrice \* num) / nn);

g.dwNum = nn;

g.dwPrice = p;

arr1[index] = g;

}

this.sendData();

this.sendStore();

}

}

break;

}

}

}

public sellGoods(id: number, num: number) {

if (num == 0) {

this.sendError(ERROR.SELL\_ZERO);

return;

}

let marketprice = this.getPriceInMarket(id);

if (marketprice == null) {

this.sendError(ERROR.MARKET\_NO\_GOOD);

return;

}

let arr = DataBase.storeGoods;

for (let i: number = 0; i < arr.length; i++) {

let good = arr[i];

if (good.dwID == id) {

DataBase.money += marketprice \* num;

good.dwNum -= num;

if (good.dwNum <= 0) {

DataBase.storeGoods.splice(i, 1);

}

this.sendData();

this.sendStore();

break;

}

}

}

public cun(num: number) {

if (num > 0 && num <= DataBase.money) {

DataBase.deposit += num;

DataBase.money -= num;

this.sendData();

}

}

public qu(num: number) {

if (num > 0 && num <= DataBase.deposit) {

DataBase.deposit -= num;

DataBase.money += num;

this.sendData();

}

}

public huan(num: number) {

if (num > 0 && num <= DataBase.money && num <= DataBase.debt) {

DataBase.debt -= num;

DataBase.money -= num;

this.sendData();

}

}

public treat(n: number) {

if (n > 0 && n < 100) {

if (n + DataBase.pow > 100) {

n = 100 - DataBase.pow;

}

let needmoney = n \* GameLogic.getInstance().data['hospital'];

if (needmoney >= DataBase.money) {

this.sendError(ERROR.MONEY\_NOT\_ENOUGH);

return;

}

DataBase.money -= needmoney;

DataBase.pow += n;

this.sendData();

}

}

public charity(n:number){

if(n > DataBase.money){

this.sendError(ERROR.MONEY\_NOT\_ENOUGH);

return;

}

let charity = GameLogic.getInstance().data['charity'];

let c:number;

if(n < charity){

let r = Math.random() \* 100;

if(r < 2){

DataBase.fame += 3;

c = 0;

}

else{

c = 1;

}

}

else{

let i = Math.floor(n / charity);

DataBase.fame += i;

c = i < 10 ? 2 : (i < 100 ? 3 : 4);

}

this.addEvent(5,1,c);

DataBase.money -= n;

this.sendData();

this.sendEvent();

}

public buyStore(price: number) {

let max = GameLogic.getInstance().data['maxstore'];

if (DataBase.maxStoreNum >= max) {

this.sendError(ERROR.MAX\_STORE\_NUM);

return;

}

let n = GameLogic.getInstance().data['storeprice'];

if (price < n) {

console.log("价格低于标准值，请勿作弊");

return;

}

if (DataBase.money < price) {

this.sendError(ERROR.MONEY\_NOT\_ENOUGH);

return;

}

else {

DataBase.maxStoreNum += 10;

if (DataBase.maxStoreNum >= max) {

DataBase.maxStoreNum = max;

}

let r = Math.floor(Math.random() \* n / 5);

DataBase.money -= (price + r);

this.sendData();

this.addEvent(5, 0, 0);

this.sendEvent();

}

}

}

declare const wx: any

class GameLogic extends egret.EventDispatcher {

public constructor() {

super();

}

private static \_instance: GameLogic;

public static getInstance(): GameLogic {

if (this.\_instance == null) {

this.\_instance = new GameLogic();

}

return this.\_instance;

}

public GameStage: egret.Stage;

public main: eui.UILayer;

public data: Object;

public goods: Object;

public strings: Object;

public cbSelected: boolean;

public openStart() {

this.initData();

this.main.removeChildren();

this.main.addChild(new StartUI());

}

private initData() {

if (this.data == null) {

this.data = RES.getRes("config\_json");

}

if (this.goods == null) {

this.goods = RES.getRes("goods\_json");

}

if (this.strings == null) {

this.strings = RES.getRes("string\_json");

}

}

public gameui: GameUI;

public startGame() {

this.main.removeChildren();

this.main.addChild(new GameUI());

}

public share(type: number) {

let wx = window["wx"];

if (wx != null) {

wx.onShareAppMessage(function () {

return {

title: '转发标题',

imageUrl: 'qua\_3\_png',

success: function (res) {

console.log('转发成功')

}

}

});

}

}

}

class varGoods {

public constructor() {

}

public dwID:number = 0;

public strName:string;

public dwPrice:number = 0;

public dwNum:number = 0;

}

class msgLifeDataRsp {

public constructor() {

}

public dwPow:number = 0;

public dwMoney:number;//Int64;

public dwDebt:number = 0;

public dwDeposit:number;//Int64;

public dwTimes:number = 0;

public dwMaxStoreNum:number = 0;

public dwFame:number = 0;

}

class msgGoodsStoreRsp {

public constructor() {

}

public goods:varGoods[] = [];

}

class msgGoodsBuyRsp {

public constructor() {

}

public goods:varGoods[] = [];

}

class Int64 {

public constructor(lowerUint: number = 0, higherUint: number = 0) {

this.\_lowValue = lowerUint;

this.\_highValue = higherUint;

}

\_highValue: number = 0;

public get higherUint(): number {

return this.\_highValue;

}

public set higherUint(value: number) {

if (this.\_highValue == value)

return;

this.\_highValue = value;

this.cacheBytes = null;

this.cacheString = [];

}

private \_lowValue: number = 0;

public get lowerUint(): number {

return this.\_lowValue;

}

public set lowerUint(value: number) {

this.\_lowValue = value;

if (this.\_lowValue == value)

return;

this.cacheBytes = null;

this.cacheString = [];

}

private cacheString: Array<string> = new Array<string>();

private cacheBytes: egret.ByteArray;

public fromString(value: string, radix: number = 10): void {

if (!value) {

this.reset();

return;

}

value = value.toLowerCase();

var div: number = 4294967296;

let low: number = 0;

var high: number = 0;

for (var i: number = 0; i < value.length; i++) {

var num: number = value.charCodeAt(i) - 48;

if (num > 9)

num -= 39;

low = low \* radix + num;

high = high \* radix + (low / div >> 0);

low = low % div;

}

this.\_lowValue = low;

this.\_highValue = high;

this.cacheString = [];

this.cacheString[radix] = value;

this.cacheBytes = null;

}

public fromBytes(bytes: egret.ByteArray, postion: number = 0): void {

try {

bytes.position = postion;

if (bytes.endian == egret.Endian.LITTLE\_ENDIAN) {

this.\_lowValue = bytes.readUnsignedInt();

this.\_highValue = bytes.readUnsignedInt();

}

else {

this.\_highValue = bytes.readUnsignedInt();

this.\_lowValue = bytes.readUnsignedInt();

}

}

catch (e) {

this.reset();

return;

}

this.cacheBytes = null;

this.cacheString = [];

}

private reset(): void {

this.\_highValue = 0;

this.\_lowValue = 0;

this.cacheBytes = null

this.cacheString = [];

}

public clone(): Int64 {

return new Int64(this.\_lowValue, this.\_highValue);

}

public copy(value: Int64): void {

this.reset();

this.\_lowValue = value.\_lowValue;

this.\_highValue = value.\_highValue;

}

public cloneTo(value: Int64): Int64 {

if (value == null) {

value = new Int64();

}

value.copy(this);

return value;

}

public equals(value: Int64): boolean {

if (value == null) return false;

return this.\_highValue == value.\_highValue && this.\_lowValue == value.\_lowValue;

}

public get bytes(): egret.ByteArray {

if (this.cacheBytes)

return this.cacheBytes;

this.cacheBytes = new egret.ByteArray();

this.cacheBytes.endian = egret.Endian.LITTLE\_ENDIAN;

this.cacheBytes.writeUnsignedInt(this.\_lowValue);

this.cacheBytes.writeUnsignedInt(this.\_highValue);

return this.cacheBytes;

}

public toNumber():number

{

var value:string = this.toString();

return value == "" ? 0 : parseInt(value);

}

public toString(radix: number = 10): string {

if (radix < 2 || radix > 36) {

throw new RangeError("基数参数必须介于 2 到 36 之间；当前值为 " + radix + "。");

}

if (this.cacheString[radix])

return this.cacheString[radix];

var result: string = "";

var lowUint: number = this.\_lowValue;

var highUint: number = this.\_highValue;

var highRemain: number;

var lowRemain: number;

var tempNum: number;

var MaxLowUint: number = Math.pow(2, 32);

while (highUint != 0 || lowUint != 0) {

highRemain = (highUint % radix);

tempNum = highRemain \* MaxLowUint + lowUint;

lowRemain = tempNum % radix;

result = lowRemain.toString(radix) + result;

highUint = (highUint - highRemain) / radix;

lowUint = (tempNum - lowRemain) / radix;

}

this.cacheString[radix] = result == "" ? "0" : result;

return this.cacheString[radix];

}

public parseData(data: egret.ByteArray): void {

this.\_highValue = data.readUnsignedInt();

this.\_lowValue = data.readUnsignedInt();

}

public toData(data: egret.ByteArray): void {

data.writeUnsignedInt(this.\_highValue);

data.writeUnsignedInt(this.\_lowValue);

}

public gc(): void {

this.cacheBytes = null;

this.cacheString = null;

}

}

class GameConst {

public constructor() {

}

}

enum ERROR {

MONEY\_NOT\_ENOUGH,

STORE\_NOT\_ENOUGH,

BUY\_ZERO,

SELL\_ZERO,

MARKET\_NO\_GOOD,

MAX\_STORE\_NUM,

NEED\_LICIENCE,

}

enum ACHIVE{

RELIVE = 1,

}

class DataBase {

public constructor() {

}

public static gameState:number;

public static gamePackage:number;

public static money:number;//Int64;

public static debt:number;

public static deposit:number;//Int64;

public static pow:number;

public static times:number;

public static maxStoreNum:number;

public static fame:number;

public static marketGoods:varGoods[];

public static storeGoods:varGoods[];

public static events:string[];

public static achives:number[];

}

declare class BaseButtonSkin extends eui.Skin{

}

declare class GameSkin extends eui.Skin{

}

declare class MarketItemSkin extends eui.Skin{

}

declare class RankSkin extends eui.Skin{

}

declare class StartSkin extends eui.Skin{

}

declare class StoreItemSkin extends eui.Skin{

}

declare type ResourceRootSelector<T extends string> = () => T;

declare type ResourceTypeSelector = (file: string) => string;

declare type ResourceNameSelector = (file: string) => string;

declare type ResourceMergerSelector = (file: string) => {

path: string;

alias: string;

};

declare module RES {

var resourceTypeSelector: ResourceTypeSelector;

var resourceNameSelector: ResourceNameSelector;

var resourceMergerSelector: ResourceMergerSelector | null;

function getResourceInfo(path: string): File | null;

function setConfigURL(url: string, root: string): void;

interface ResourceInfo {

url: string;

type: string;

root: string;

crc32?: string;

size?: number;

name: string;

soundType?: string;

scale9grid?: string;

groupNames?: string[];

extra?: boolean;

promise?: Promise<any>;

}

interface Data {

resourceRoot: string;

typeSelector: ResourceTypeSelector;

mergeSelector: ResourceMergerSelector | null;

fileSystem: FileSystem;

groups: {

[groupName: string]: string[];

};

alias: {

[aliasName: string]: string;

};

}

class ResourceConfig {

config: Data;

constructor();

init(): Promise<void>;

\_\_temp\_\_get\_\_type\_\_via\_\_url(url\_or\_alias: string): string;

getKeyByAlias(aliasName: string): string;

createGroup(name: string, keys: Array<string>, override?: boolean): boolean;

addSubkey(subkey: string, name: string): void;

addAlias(alias: any, key: any): void;

getType(key: string): string;

addResourceData(data: {

name: string;

type?: string;

url: string;

root?: string;

}): void;

destory(): void;

}

}

declare module RES {

class ResourceLoader {

private groupTotalDic;

private numLoadedDic;

private itemListDic;

private groupErrorDic;

private retryTimesDic;

maxRetryTimes: number;

private priorityQueue;

private reporterDic;

private dispatcherDic;

private failedList;

private loadItemErrorDic;

private errorDic;

load(list: ResourceInfo[], groupName: string, priority: number, reporter?: PromiseTaskReporter): Promise<any>;

private loadingCount;

thread: number;

private next();

private removeGroupName(groupName);

private queueIndex;

private getOneResourceInfo();

loadResource(r: ResourceInfo, p?: RES.processor.Processor): Promise<any>;

unloadResource(r: ResourceInfo): Promise<any>;

}

}

declare module RES {

var systemPid: number;

let checkCancelation: MethodDecorator;

function profile(): void;

var host: ProcessHost;

var config: ResourceConfig;

var queue: ResourceLoader;

interface ProcessHost {

state: {

[index: string]: number;

};

resourceConfig: ResourceConfig;

load: (resource: ResourceInfo, processor?: string | processor.Processor) => Promise<any>;

unload: (resource: ResourceInfo) => Promise<any>;

save: (rexource: ResourceInfo, data: any) => void;

get: (resource: ResourceInfo) => any;

remove: (resource: ResourceInfo) => void;

}

class ResourceManagerError extends Error {

static errorMessage: {

1001: string;

1002: string;

1005: string;

2001: string;

2002: string;

2003: string;

2004: string;

2005: string;

2006: string;

};

private \_\_resource\_manager\_error\_\_;

constructor(code: number, replacer?: Object, replacer2?: Object);

}

}

declare namespace RES {

interface PromiseTaskReporter {

onProgress?: (current: number, total: number) => void;

onCancel?: () => void;

}

}

declare module RES {

let checkNull: MethodDecorator;

let FEATURE\_FLAG: {

FIX\_DUPLICATE\_LOAD: number;

};

namespace upgrade {

function setUpgradeGuideLevel(level: "warning" | "silent"): void;

}

}

declare module RES.processor {

interface Processor {

onLoadStart(host: ProcessHost, resource: ResourceInfo): Promise<any>;

onRemoveStart(host: ProcessHost, resource: ResourceInfo): Promise<any>;

getData?(host: ProcessHost, resource: ResourceInfo, key: string, subkey: string): any;

}

function isSupport(resource: ResourceInfo): Processor;

function map(type: string, processor: Processor): void;

function getRelativePath(url: string, file: string): string;

var ImageProcessor: Processor;

var BinaryProcessor: Processor;

var TextProcessor: Processor;

var JsonProcessor: Processor;

var XMLProcessor: Processor;

var CommonJSProcessor: Processor;

const SheetProcessor: Processor;

var FontProcessor: Processor;

var SoundProcessor: Processor;

var MovieClipProcessor: Processor;

const MergeJSONProcessor: Processor;

const ResourceConfigProcessor: Processor;

const LegacyResourceConfigProcessor: Processor;

var PVRProcessor: Processor;

const \_map: {

[index: string]: Processor;

};

}

declare module RES {

interface File {

url: string;

type: string;

name: string;

root: string;

}

interface Dictionary {

[file: string]: File | Dictionary;

}

interface FileSystem {

addFile(filename: string, type?: string, root?: string): any;

getFile(filename: string): File | null;

profile(): void;

}

class NewFileSystem {

private data;

constructor(data: Dictionary);

profile(): void;

addFile(filename: string, type?: string): void;

getFile(filename: string): File | null;

private reslove(dirpath);

private mkdir(dirpath);

private exists(dirpath);

}

var fileSystem: FileSystem;

}

declare module RES {

class ResourceEvent extends egret.Event {

static ITEM\_LOAD\_ERROR: string;

static CONFIG\_COMPLETE: string;

static CONFIG\_LOAD\_ERROR: string;

static GROUP\_PROGRESS: string;

static GROUP\_COMPLETE: string;

static GROUP\_LOAD\_ERROR: string;

constructor(type: string, bubbles?: boolean, cancelable?: boolean);

itemsLoaded: number;

itemsTotal: number;

groupName: string;

resItem: ResourceItem;

}

}

declare module RES {

namespace ResourceItem {

const TYPE\_XML: string;

const TYPE\_IMAGE: string;

const TYPE\_BIN: string;

const TYPE\_TEXT: string;

const TYPE\_JSON: string;

const TYPE\_SHEET: string;

const TYPE\_FONT: string;

const TYPE\_SOUND: string;

function convertToResItem(r: ResourceInfo): ResourceItem;

}

interface ResourceItem extends ResourceInfo {

name: string;

url: string;

type: string;

data: ResourceInfo;

crc32?: string;

size?: number;

soundType?: string;

}

}

declare namespace RES {

namespace path {

const normalize: (filename: string) => string;

const basename: (filename: string) => string;

const dirname: (path: string) => string;

}

}

declare namespace RES {

}

declare module RES {

type GetResAsyncCallback = (value?: any, key?: string) => any;

function registerAnalyzer(type: string, analyzerClass: any): void;

function loadConfig(url: string, resourceRoot: string): Promise<void>;

function loadGroup(name: string, priority?: number, reporter?: PromiseTaskReporter): Promise<void>;

function isGroupLoaded(name: string): boolean;

function getGroupByName(name: string): Array<ResourceItem>;

function createGroup(name: string, keys: Array<string>, override?: boolean): boolean;

function hasRes(key: string): boolean;

function getResAsync(key: string): Promise<any>;

function getResAsync(key: string, compFunc: GetResAsyncCallback, thisObject: any): void;

function getResByUrl(url: string, compFunc: Function, thisObject: any, type?: string): void;

function destroyRes(name: string, force?: boolean): Promise<boolean>;

function setMaxLoadingThread(thread: number): void;

function setMaxRetryTimes(retry: number): void;

function addEventListener(type: string, listener: (event: egret.Event) => void, thisObject: any, useCapture?: boolean, priority?: number): void;

function removeEventListener(type: string, listener: (event: egret.Event) => void, thisObject: any, useCapture?: boolean): void;

function $addResourceData(data: {

name: string;

type: string;

url: string;

}): void;

class Resource extends egret.EventDispatcher {

loadConfig(): Promise<void>;

isGroupLoaded(name: string): boolean;

getGroupByName(name: string): Array<ResourceInfo>;

loadGroup(name: string, priority?: number, reporter?: PromiseTaskReporter): Promise<any>;

private \_loadGroup(name, priority?, reporter?);

loadResources(keys: string[], reporter?: PromiseTaskReporter): Promise<any>;

createGroup(name: string, keys: Array<string>, override?: boolean): boolean;

hasRes(key: string): boolean;

getRes(resKey: string): any;

getResAsync(key: string): Promise<any>;

getResAsync(key: string, compFunc: GetResAsyncCallback, thisObject: any): void;

getResByUrl(url: string, compFunc: Function, thisObject: any, type?: string): Promise<any> | void;

destroyRes(name: string, force?: boolean): Promise<boolean>;

setMaxLoadingThread(thread: number): void;

setMaxRetryTimes(retry: number): void;

addResourceData(data: {

name: string;

type: string;

url: string;

}): void;

}

}

class StringUtil {

public constructor() {

}

public static getSwfLangTextFlowVar(StrID: string, valArr: string[]): egret.ITextElement[] {

return new egret.HtmlTextParser().parser(StringUtil.getSwfLangStrVar(StrID, valArr));

}

public static getSwfLangStrVarByID(StrID: string, valArr: string[]): string {

if (GameLogic.getInstance().strings == null) {

return StrID;

}

var data: any = GameLogic.getInstance().strings[StrID];

if (data == null) {

return StrID;

}

return StringUtil.getSwfLangStrVar(data,valArr);

}

public static getSwfLangStrVar(strData: string, valArr: string[]): string {

var indexpre: number;

var indexback: number;

var strget: string;

indexpre = strData.indexOf("{");

indexback = strData.indexOf("}");

var nextOffset: number = 0;

var firstIndex: number;

var strFlagPre: number;

var strFlagBack: number;

var strFlag: string;

while (indexpre != -1 && indexback != -1) {

strget = strData.substring(indexpre, indexback + 1);

firstIndex = strData.indexOf("@", nextOffset);

//var number: int = int(strData.charAt(strData.indexOf("@", nextOffset) + 1));

var numberic: number = parseInt(strData.substring(firstIndex + 1, strData.indexOf(":", firstIndex))) - 1;

if (numberic == NaN) {

return "stringError:" + strData;

}

strFlagPre = strData.indexOf("!#[", nextOffset) + 3;

if (strFlagPre > 2) {

strFlagBack = strData.indexOf("]@", nextOffset);

strFlag = strData.substring(strFlagPre, strFlagBack);

valArr[numberic] = StringUtil.getSwfLangStr(strFlag + valArr[numberic]);

}

var strreplace: string = valArr[numberic].toString();

strData = strData.replace(strget, strreplace);

nextOffset = indexpre + strreplace.length;

indexpre = strData.indexOf("{", nextOffset);

indexback = strData.indexOf("}", nextOffset);

}

return strData;

}

public static getSwfLangStr(StrID: string): string {

if (GameLogic.getInstance().strings == null) {

return StrID;

}

var data: any = GameLogic.getInstance().strings[StrID];

if (data == null) {

return StrID;

}

return data.toString();

}

}

declare namespace egret {

class Ease {

constructor();

static get(amount: number): (t: number) => number;

static getPowIn(pow: number): (t: number) => number;

static getPowOut(pow: number): (t: number) => number;

static getPowInOut(pow: number): (t: number) => number;

static quadIn: (t: number) => number;

static quadOut: (t: number) => number;

static quadInOut: (t: number) => number;

static cubicIn: (t: number) => number;

static cubicOut: (t: number) => number;

static cubicInOut: (t: number) => number;

static quartIn: (t: number) => number;

static quartOut: (t: number) => number;

static quartInOut: (t: number) => number;

static quintIn: (t: number) => number;

static quintOut: (t: number) => number;

static quintInOut: (t: number) => number;

static sineIn(t: number): number;

static sineOut(t: number): number;

static sineInOut(t: number): number;

static getBackIn(amount: number): (t: number) => number;

static backIn: (t: number) => number;

static getBackOut(amount: number): (t: any) => number;

static backOut: (t: any) => number;

static getBackInOut(amount: number): (t: number) => number;

static backInOut: (t: number) => number;

static circIn(t: number): number;

static circOut(t: number): number;

static circInOut(t: number): number;

static bounceIn(t: number): number;

static bounceOut(t: number): number;

static bounceInOut(t: number): number;

static getElasticIn(amplitude: number, period: number): (t: number) => number;

static elasticIn: (t: number) => number;

static getElasticOut(amplitude: number, period: number): (t: number) => number;

static elasticOut: (t: number) => number;

static getElasticInOut(amplitude: number, period: number): (t: number) => number;

static elasticInOut: (t: number) => number;

}

}

declare namespace egret {

class Tween extends EventDispatcher {

private static NONE;

private static LOOP;

private static REVERSE;

private static \_tweens;

private static IGNORE;

private static \_plugins;

private static \_inited;

private \_target;

private \_useTicks;

private ignoreGlobalPause;

private loop;

private pluginData;

private \_curQueueProps;

private \_initQueueProps;

private \_steps;

private paused;

private duration;

private \_prevPos;

private position;

private \_prevPosition;

private \_stepPosition;

private passive;

static get(target: any, props?: {

loop?: boolean;

onChange?: Function;

onChangeObj?: any;

}, pluginData?: any, override?: boolean): Tween;

static removeTweens(target: any): void;

static pauseTweens(target: any): void;

static resumeTweens(target: any): void;

private static tick(timeStamp, paused?);

private static \_lastTime;

private static \_register(tween, value);

static removeAllTweens(): void;

constructor(target: any, props: any, pluginData: any);

private initialize(target, props, pluginData);

setPosition(value: number, actionsMode?: number): boolean;

private \_runAction(action, startPos, endPos, includeStart?);

private \_updateTargetProps(step, ratio);

setPaused(value: boolean): Tween;

private \_cloneProps(props);

private \_addStep(o);

private \_appendQueueProps(o);

private \_addAction(o);

private \_set(props, o);

wait(duration: number, passive?: boolean): Tween;

to(props: any, duration?: number, ease?: Function): Tween;

call(callback: Function, thisObj?: any, params?: any[]): Tween;

set(props: any, target?: any): Tween;

play(tween?: Tween): Tween;

pause(tween?: Tween): Tween;

$tick(delta: number): void;

}

}

declare namespace egret.tween {

type EaseType = 'quadIn' | 'quadOut' | 'quadOut' | 'quadInOut' | 'cubicIn' | 'cubicOut' | 'cubicInOut' | 'quartIn' | 'quartOut' | 'quartInOut' | 'quintIn' | 'quintOut' | 'quintInOut' | 'sineIn' | 'sineOut' | 'sineInOut' | 'backIn' | 'backOut' | 'backInOut' | 'circIn' | 'circOut' | 'circInOut' | 'bounceIn' | 'bounceOut' | 'bounceInOut' | 'elasticIn' | 'elasticOut' | 'elasticInOut';

abstract class BasePath extends EventDispatcher {

name: string;

}

class To extends BasePath {

props: Object;

duration: number;

ease: EaseType | Function;

}

class Wait extends BasePath {

duration: number;

passive: boolean;

}

class Set extends BasePath {

props: Object;

}

class Tick extends BasePath {

delta: number;

}

class TweenItem extends EventDispatcher {

private tween;

constructor();

private \_props;

props: any;

private \_target;

target: any;

private \_paths;

paths: BasePath[];

play(position?: number): void;

pause(): void;

private isStop;

stop(): void;

private createTween(position);

private applyPaths();

private applyPath(path);

private pathComplete(path);

}

class TweenGroup extends EventDispatcher {

private completeCount;

constructor();

private \_items;

items: TweenItem[];

private registerEvent(add);

play(time?: number): void;

pause(): void;

stop(): void;

private itemComplete(e);

}

}

declare var global: any;

declare var \_\_global: any;

declare let \_\_define: any;

declare namespace egret {

interface IHashObject {

hashCode: number;

}

let $hashCount: number;

class HashObject implements IHashObject {

readonly hashCode: number;

}

}

declare namespace egret {

class EventDispatcher extends HashObject implements IEventDispatcher {

constructor(target?: IEventDispatcher);

$EventDispatcher: Object;

$getEventMap(useCapture?: boolean): any;

addEventListener(type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?: number): void;

once(type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?: number): void;

$addListener(type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?: number, dispatchOnce?: boolean): void;

$insertEventBin(list: any[], type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?: number, dispatchOnce?: boolean): boolean;

removeEventListener(type: string, listener: Function, thisObject: any, useCapture?: boolean): void;

$removeEventBin(list: any[], listener: Function, thisObject: any): boolean;

hasEventListener(type: string): boolean;

willTrigger(type: string): boolean;

dispatchEvent(event: Event): boolean;

$notifyListener(event: Event, capturePhase: boolean): boolean;

dispatchEventWith(type: string, bubbles?: boolean, data?: any, cancelable?: boolean): boolean;

}

}

declare namespace egret.sys {

interface EventBin {

type: string;

listener: Function;

thisObject: any;

priority: number;

target: IEventDispatcher;

useCapture: boolean;

dispatchOnce: boolean;

}

}

declare namespace egret {

class Filter extends HashObject {

type: string;

$id: number;

$uniforms: any;

protected paddingTop: number;

protected paddingBottom: number;

protected paddingLeft: number;

protected paddingRight: number;

$obj: any;

constructor();

$toJson(): string;

protected updatePadding(): void;

onPropertyChange(): void;

}

}

declare namespace egret {

const enum RenderMode {

NONE = 1,

FILTER = 2,

CLIP = 3,

SCROLLRECT = 4,

}

class DisplayObject extends EventDispatcher {

constructor();

$nativeDisplayObject: egret\_native.NativeDisplayObject;

protected createNativeDisplayObject(): void;

$hasAddToStage: boolean;

$children: DisplayObject[];

private $name;

name: string;

$parent: DisplayObjectContainer;

readonly parent: DisplayObjectContainer;

$setParent(parent: DisplayObjectContainer): void;

$onAddToStage(stage: Stage, nestLevel: number): void;

$onRemoveFromStage(): void;

$stage: Stage;

$nestLevel: number;

$useTranslate: boolean;

protected $updateUseTransform(): void;

readonly stage: Stage;

matrix: Matrix;

private $matrix;

private $matrixDirty;

$getMatrix(): Matrix;

$setMatrix(matrix: Matrix, needUpdateProperties?: boolean): void;

private $concatenatedMatrix;

$getConcatenatedMatrix(): Matrix;

private $invertedConcatenatedMatrix;

$getInvertedConcatenatedMatrix(): Matrix;

$x: number;

x: number;

$getX(): number;

$setX(value: number): boolean;

$y: number;

y: number;

$getY(): number;

$setY(value: number): boolean;

private $scaleX;

scaleX: number;

$getScaleX(): number;

$setScaleX(value: number): void;

private $scaleY;

scaleY: number;

$getScaleY(): number;

$setScaleY(value: number): void;

private $rotation;

rotation: number;

$getRotation(): number;

$setRotation(value: number): void;

private $skewX;

private $skewXdeg;

skewX: number;

$setSkewX(value: number): void;

private $skewY;

private $skewYdeg;

skewY: number;

$setSkewY(value: number): void;

width: number;

$getWidth(): number;

$explicitWidth: number;

$setWidth(value: number): void;

height: number;

$explicitHeight: number;

$getHeight(): number;

$setHeight(value: number): void;

readonly measuredWidth: number;

readonly measuredHeight: number;

$anchorOffsetX: number;

anchorOffsetX: number;

$setAnchorOffsetX(value: number): void;

$anchorOffsetY: number;

anchorOffsetY: number;

$setAnchorOffsetY(value: number): void;

$visible: boolean;

visible: boolean;

$setVisible(value: boolean): void;

$displayList: egret.sys.DisplayList;

private $cacheAsBitmap;

cacheAsBitmap: boolean;

$setHasDisplayList(value: boolean): void;

$cacheDirty: boolean;

$cacheDirtyUp(): void;

$alpha: number;

alpha: number;

$setAlpha(value: number): void;

static defaultTouchEnabled: boolean;

$touchEnabled: boolean;

touchEnabled: boolean;

$getTouchEnabled(): boolean;

$setTouchEnabled(value: boolean): void;

$scrollRect: Rectangle;

scrollRect: Rectangle;

mask: DisplayObject | Rectangle;

private $setMaskRect(value);

$filters: Array<Filter | CustomFilter>;

filters: Array<Filter | CustomFilter>;

getTransformedBounds(targetCoordinateSpace: DisplayObject, resultRect?: Rectangle): Rectangle;

getBounds(resultRect?: Rectangle, calculateAnchor?: boolean): egret.Rectangle;

$getTransformedBounds(targetCoordinateSpace: DisplayObject, resultRect?: Rectangle): Rectangle;

globalToLocal(stageX?: number, stageY?: number, resultPoint?: Point): Point;

localToGlobal(localX?: number, localY?: number, resultPoint?: Point): Point;

$getOriginalBounds(): Rectangle;

$measureChildBounds(bounds: Rectangle): void;

$getContentBounds(): Rectangle;

$measureContentBounds(bounds: Rectangle): void;

$parentDisplayList: egret.sys.DisplayList;

$renderNode: sys.RenderNode;

$renderDirty: boolean;

$getRenderNode(): sys.RenderNode;

private updateRenderMode();

$renderMode: RenderMode;

private $measureFiltersOffset(fromParent);

$getConcatenatedMatrixAt(root: DisplayObject, matrix: Matrix): void;

$updateRenderNode(): void;

hitTestPoint(x: number, y: number, shapeFlag?: boolean): boolean;

$addListener(type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?: number, dispatchOnce?: boolean): void;

removeEventListener(type: string, listener: Function, thisObject: any, useCapture?: boolean): void;

$getPropagationList(target: DisplayObject): DisplayObject[];

$dispatchPropagationEvent(event: Event, list: DisplayObject[], targetIndex: number): void;

willTrigger(type: string): boolean;

}

}

declare namespace egret {

let $TextureScaleFactor: number;

class Texture extends HashObject {

constructor();

private $textureWidth;

readonly textureWidth: number;

$getTextureWidth(): number;

readonly textureHeight: number;

$getTextureHeight(): number;

$getScaleBitmapWidth(): number;

$getScaleBitmapHeight(): number;

$initData(bitmapX: number, bitmapY: number, bitmapWidth: number, bitmapHeight: number, offsetX: number, offsetY: number, textureWidth: number, textureHeight: number, sourceWidth: number, sourceHeight: number, rotated?: boolean): void;

getPixel32(x: number, y: number): number[];

getPixels(x: number, y: number, width?: number, height?: number): number[];

toDataURL(type: string, rect?: egret.Rectangle, encoderOptions?: any): string;

dispose(): void;

}

}

declare namespace egret {

class Event extends HashObject {

static ADDED\_TO\_STAGE: string;

static REMOVED\_FROM\_STAGE: string;

static ENTER\_FRAME: string;

static LOOP\_COMPLETE: string;

readonly target: any;

$setTarget(target: any): boolean;

$isPropagationImmediateStopped: boolean;

static dispatchEvent(target: IEventDispatcher, type: string, bubbles?: boolean, data?: any): boolean;

static \_getPropertyData(EventClass: any): any;

static create<T extends Event>(EventClass: {

new (type: string, bubbles?: boolean, cancelable?: boolean): T;

eventPool?: Event[];

}, type: string, bubbles?: boolean, cancelable?: boolean): T;

static release(event: Event): void;

}

}

declare let RELEASE: boolean;

declare namespace egret {

function $error(code: number, ...params: any[]): void;

function $warn(code: number, ...params: any[]): void;

function getString(code: number, ...params: any[]): string;

function $markCannotUse(instance: any, property: string, defaultVale: any): void;

}

declare namespace egret {

class Point extends HashObject {

static release(point: Point): void;

constructor(x?: number, y?: number);

let $TempPoint: Point;

}

declare namespace egret {

class DisplayObjectContainer extends DisplayObject {

static $EVENT\_ADD\_TO\_STAGE\_LIST: DisplayObject[];

static $EVENT\_REMOVE\_FROM\_STAGE\_LIST: DisplayObject[];

constructor();

addChild(child: DisplayObject): DisplayObject;

addChildAt(child: DisplayObject, index: number): DisplayObject;

$doAddChild(child: DisplayObject, index: number, notifyListeners?: boolean): DisplayObject;

contains(child: DisplayObject): boolean;

getChildAt(index: number): DisplayObject;

getChildIndex(child: egret.DisplayObject): number;

getChildByName(name: string): DisplayObject;

$doRemoveChild(index: number, notifyListeners?: boolean): DisplayObject;

setChildIndex(child: DisplayObject, index: number): void;

private doSetChildIndex(child, index);

$measureChildBounds(bounds: Rectangle): void;

$touchChildren: boolean;

$hitTest(stageX: number, stageY: number): DisplayObject;

}

}

var \_\_reflect = (this && this.\_\_reflect) || function (p, c, t) {

p.\_\_class\_\_ = c, t ? t.push(c) : t = [c], p.\_\_types\_\_ = p.\_\_types\_\_ ? t.concat(p.\_\_types\_\_) : t;

};

var \_\_extends = this && this.\_\_extends || function \_\_extends(t, e) {

function r() {

this.constructor = t;

}

for (var i in e) e.hasOwnProperty(i) && (t[i] = e[i]);

r.prototype = e.prototype, t.prototype = new r();

};

if (typeof global == 'undefined') {

var global = window;

}

if (typeof \_\_global == 'undefined') {

var \_\_global = global;

}

var \_\_define = this && this.\_\_define || function (o, p, g, s) { Object.defineProperty(o, p, { configurable: true, enumerable: true, get: g, set: s }); };

var egret;

(function (egret) {

function HashObject() {

this.$hashCode = egret.$hashCount++;

}

Object.defineProperty(HashObject.prototype, "hashCode", {

get: function () {

return this.$hashCode;

},

enumerable: true,

configurable: true

});

return HashObject;

}());

egret.HashObject = HashObject;

\_\_reflect(HashObject.prototype, "egret.HashObject", ["egret.IHashObject"]);

})(egret || (egret = {}));

var egret;

(function (egret) {

var ONCE\_EVENT\_LIST = [];

var EventDispatcher = (function (\_super) {

\_\_extends(EventDispatcher, \_super);

function EventDispatcher(target) {

if (target === void 0) { target = null; }

var \_this = \_super.call(this) || this;

\_this.$EventDispatcher = {

0: target ? target : \_this,

1: {},

2: {},

3: 0

};

return \_this;

}

EventDispatcher.prototype.$getEventMap = function (useCapture) {

var values = this.$EventDispatcher;

var eventMap = useCapture ? values[2 /\* captureEventsMap \*/] : values[1 /\* eventsMap \*/];

return eventMap;

};

EventDispatcher.prototype.addEventListener = function (type, listener, thisObject, useCapture, priority) {

this.$addListener(type, listener, thisObject, useCapture, priority);

};

EventDispatcher.prototype.once = function (type, listener, thisObject, useCapture, priority) {

this.$addListener(type, listener, thisObject, useCapture, priority, true);

};

EventDispatcher.prototype.$addListener = function (type, listener, thisObject, useCapture, priority, dispatchOnce) {

if (true && !listener) {

egret.$error(1003, "listener");

}

var values = this.$EventDispatcher;

var eventMap = useCapture ? values[2 /\* captureEventsMap \*/] : values[1 /\* eventsMap \*/];

var list = eventMap[type];

if (!list) {

list = eventMap[type] = [];

}

else if (values[3 /\* notifyLevel \*/] !== 0) {

eventMap[type] = list = list.concat();

}

this.$insertEventBin(list, type, listener, thisObject, useCapture, priority, dispatchOnce);

};

EventDispatcher.prototype.$insertEventBin = function (list, type, listener, thisObject, useCapture, priority, dispatchOnce) {

priority = +priority | 0;

var insertIndex = -1;

var length = list.length;

for (var i = 0; i < length; i++) {

var bin = list[i];

if (bin.listener == listener && bin.thisObject == thisObject && bin.target == this) {

return false;

}

if (insertIndex == -1 && bin.priority < priority) {

insertIndex = i;

}

}

var eventBin = {

type: type, listener: listener, thisObject: thisObject, priority: priority,

target: this, useCapture: useCapture, dispatchOnce: !!dispatchOnce

};

if (insertIndex !== -1) {

list.splice(insertIndex, 0, eventBin);

}

else {

list.push(eventBin);

}

return true;

};

EventDispatcher.prototype.removeEventListener = function (type, listener, thisObject, useCapture) {

var values = this.$EventDispatcher;

var eventMap = useCapture ? values[2 /\* captureEventsMap \*/] : values[1 /\* eventsMap \*/];

var list = eventMap[type];

if (!list) {

return;

}

if (values[3 /\* notifyLevel \*/] !== 0) {

eventMap[type] = list = list.concat();

}

this.$removeEventBin(list, listener, thisObject);

if (list.length == 0) {

eventMap[type] = null;

}

};

EventDispatcher.prototype.$removeEventBin = function (list, listener, thisObject) {

var length = list.length;

for (var i = 0; i < length; i++) {

var bin = list[i];

if (bin.listener == listener && bin.thisObject == thisObject && bin.target == this) {

list.splice(i, 1);

return true;

}

}

return false;

};

EventDispatcher.prototype.hasEventListener = function (type) {

var values = this.$EventDispatcher;

return !!(values[1 /\* eventsMap \*/][type] || values[2 /\* captureEventsMap \*/][type]);

};

EventDispatcher.prototype.willTrigger = function (type) {

return this.hasEventListener(type);

};

EventDispatcher.prototype.dispatchEvent = function (event) {

event.$currentTarget = this.$EventDispatcher[0 /\* eventTarget \*/];

event.$setTarget(event.$currentTarget);

return this.$notifyListener(event, false);

};

EventDispatcher.prototype.$notifyListener = function (event, capturePhase) {

var values = this.$EventDispatcher;

var eventMap = capturePhase ? values[2 /\* captureEventsMap \*/] : values[1 /\* eventsMap \*/];

var list = eventMap[event.$type];

if (!list) {

return true;

}

var length = list.length;

if (length == 0) {

return true;

}

var onceList = ONCE\_EVENT\_LIST;

values[3 /\* notifyLevel \*/]++;

for (var i = 0; i < length; i++) {

var eventBin = list[i];

eventBin.listener.call(eventBin.thisObject, event);

if (eventBin.dispatchOnce) {

onceList.push(eventBin);

}

if (event.$isPropagationImmediateStopped) {

break;

}

}

values[3 /\* notifyLevel \*/]--;

while (onceList.length) {

var eventBin = onceList.pop();

eventBin.target.removeEventListener(eventBin.type, eventBin.listener, eventBin.thisObject, eventBin.useCapture);

}

return !event.$isDefaultPrevented;

};

EventDispatcher.prototype.dispatchEventWith = function (type, bubbles, data, cancelable) {

if (bubbles || this.hasEventListener(type)) {

var event\_1 = egret.Event.create(egret.Event, type, bubbles, cancelable);

event\_1.data = data;

var result = this.dispatchEvent(event\_1);

egret.Event.release(event\_1);

return result;

}

return true;

};

return EventDispatcher;

}(egret.HashObject));

egret.EventDispatcher = EventDispatcher;

\_\_reflect(EventDispatcher.prototype, "egret.EventDispatcher", ["egret.IEventDispatcher"]);

})(egret || (egret = {}));

var egret;

(function (egret) {

var Filter = (function (\_super) {

\_\_extends(Filter, \_super);

function Filter() {

var \_this = \_super.call(this) || this;

\_this.type = null;

\_this.$id = null;

\_this.paddingTop = 0;

\_this.paddingBottom = 0;

\_this.paddingLeft = 0;

\_this.paddingRight = 0;

\_this.$uniforms = {};

if (egret.nativeRender) {

egret\_native.NativeDisplayObject.createFilter(\_this);

}

return \_this;

}

Filter.prototype.$toJson = function () {

return '';

};

Filter.prototype.updatePadding = function () {

};

Filter.prototype.onPropertyChange = function () {

var self = this;

self.updatePadding();

if (egret.nativeRender) {

egret\_native.NativeDisplayObject.setFilterPadding(self.$id, self.paddingTop, self.paddingBottom, self.paddingLeft, self.paddingRight);

egret\_native.NativeDisplayObject.setDataToFilter(self);

}

};

return Filter;

}(egret.HashObject));

egret.Filter = Filter;

\_\_reflect(Filter.prototype, "egret.Filter");

})(egret || (egret = {}));

var egret;

(function (egret) {

function clampRotation(value) {

value %= 360;

if (value > 180) {

value -= 360;

}

else if (value < -180) {

value += 360;

}

return value;

}

var DisplayObject = (function (\_super) {

\_\_extends(DisplayObject, \_super);

function DisplayObject() {

var \_this = \_super.call(this) || this;

\_this.$children = null;

\_this.$name = "";

\_this.$parent = null;

\_this.$stage = null;

\_this.$nestLevel = 0;

\_this.$useTranslate = false;

\_this.$matrix = new egret.Matrix();

\_this.$matrixDirty = false;

\_this.$x = 0;

\_this.$y = 0;

\_this.$scaleX = 1;

\_this.$scaleY = 1;

\_this.$rotation = 0;

\_this.$skewX = 0;

\_this.$skewXdeg = 0;

\_this.$skewY = 0;

\_this.$skewYdeg = 0;

\_this.$explicitWidth = NaN;

\_this.$explicitHeight = NaN;

\_this.$anchorOffsetX = 0;

\_this.$anchorOffsetY = 0;

\_this.$visible = true;

\_this.$displayList = null;

\_this.$cacheAsBitmap = false;

\_this.$cacheDirty = false;

\_this.$alpha = 1;

\_this.$touchEnabled = DisplayObject.defaultTouchEnabled;

\_this.$scrollRect = null;

\_this.$blendMode = 0;

\_this.$maskedObject = null;

\_this.$mask = null;

\_this.$parentDisplayList = null;

\_this.$renderNode = null;

\_this.$renderDirty = false;

\_this.$renderMode = null;

if (egret.nativeRender) {

\_this.createNativeDisplayObject();

}

return \_this;

}

DisplayObject.prototype.createNativeDisplayObject = function () {

this.$nativeDisplayObject = new egret\_native.NativeDisplayObject(0 /\* CONTAINER \*/);

};

Object.defineProperty(DisplayObject.prototype, "name", {

get: function () {

return this.$name;

},

set: function (value) {

this.$name = value;

},

enumerable: true,

configurable: true

});

Object.defineProperty(DisplayObject.prototype, "parent", {

get: function () {

return this.$parent;

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setParent = function (parent) {

this.$parent = parent;

};

DisplayObject.prototype.$onAddToStage = function (stage, nestLevel) {

var self = this;

self.$stage = stage;

self.$nestLevel = nestLevel;

self.$hasAddToStage = true;

egret.Sprite.$EVENT\_ADD\_TO\_STAGE\_LIST.push(self);

};

DisplayObject.prototype.$onRemoveFromStage = function () {

var self = this;

self.$nestLevel = 0;

egret.Sprite.$EVENT\_REMOVE\_FROM\_STAGE\_LIST.push(self);

};

DisplayObject.prototype.$updateUseTransform = function () {

var self = this;

if (self.$scaleX == 1 && self.$scaleY == 1 && self.$skewX == 0 && self.$skewY == 0) {

self.$useTranslate = false;

}

else {

self.$useTranslate = true;

}

};

Object.defineProperty(DisplayObject.prototype, "stage", {

get: function () {

return this.$stage;

},

enumerable: true,

configurable: true

});

Object.defineProperty(DisplayObject.prototype, "matrix", {

get: function () {

return this.$getMatrix().clone();

},

set: function (value) {

this.$setMatrix(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getMatrix = function () {

var self = this;

if (self.$matrixDirty) {

self.$matrixDirty = false;

self.$matrix.$updateScaleAndRotation(self.$scaleX, self.$scaleY, self.$skewX, self.$skewY);

}

self.$matrix.tx = self.$x;

self.$matrix.ty = self.$y;

return self.$matrix;

};

DisplayObject.prototype.$setMatrix = function (matrix, needUpdateProperties) {

if (needUpdateProperties === void 0) { needUpdateProperties = true; }

var self = this;

var m = self.$matrix;

m.a = matrix.a;

m.b = matrix.b;

m.c = matrix.c;

m.d = matrix.d;

self.$x = matrix.tx;

self.$y = matrix.ty;

self.$matrixDirty = false;

if (m.a == 1 && m.b == 0 && m.c == 0 && m.d == 1) {

self.$useTranslate = false;

}

else {

self.$useTranslate = true;

}

if (needUpdateProperties) {

self.$scaleX = m.$getScaleX();

self.$scaleY = m.$getScaleY();

self.$skewX = matrix.$getSkewX();

self.$skewY = matrix.$getSkewY();

self.$skewXdeg = clampRotation(self.$skewX \* 180 / Math.PI);

self.$skewYdeg = clampRotation(self.$skewY \* 180 / Math.PI);

self.$rotation = clampRotation(self.$skewY \* 180 / Math.PI);

}

if (egret.nativeRender) {

self.$nativeDisplayObject.setMatrix(matrix.a, matrix.b, matrix.c, matrix.d, matrix.tx, matrix.ty);

}

};

DisplayObject.prototype.$getConcatenatedMatrix = function () {

var self = this;

var matrix = self.$concatenatedMatrix;

if (!matrix) {

matrix = self.$concatenatedMatrix = new egret.Matrix();

}

if (self.$parent) {

self.$parent.$getConcatenatedMatrix().$preMultiplyInto(self.$getMatrix(), matrix);

}

else {

matrix.copyFrom(self.$getMatrix());

}

var offsetX = self.$anchorOffsetX;

var offsetY = self.$anchorOffsetY;

var rect = self.$scrollRect;

if (rect) {

matrix.$preMultiplyInto(egret.$TempMatrix.setTo(1, 0, 0, 1, -rect.x - offsetX, -rect.y - offsetY), matrix);

}

else if (offsetX != 0 || offsetY != 0) {

matrix.$preMultiplyInto(egret.$TempMatrix.setTo(1, 0, 0, 1, -offsetX, -offsetY), matrix);

}

return self.$concatenatedMatrix;

};

DisplayObject.prototype.$getInvertedConcatenatedMatrix = function () {

var self = this;

if (!self.$invertedConcatenatedMatrix) {

self.$invertedConcatenatedMatrix = new egret.Matrix();

}

self.$getConcatenatedMatrix().$invertInto(self.$invertedConcatenatedMatrix);

return self.$invertedConcatenatedMatrix;

};

Object.defineProperty(DisplayObject.prototype, "x", {

get: function () {

return this.$getX();

},

set: function (value) {

this.$setX(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getX = function () {

return this.$x;

};

DisplayObject.prototype.$setX = function (value) {

var self = this;

if (self.$x == value) {

return false;

}

self.$x = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setX(value);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

return true;

};

Object.defineProperty(DisplayObject.prototype, "y", {

get: function () {

return this.$getY();

},

set: function (value) {

this.$setY(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getY = function () {

return this.$y;

};

DisplayObject.prototype.$setY = function (value) {

var self = this;

if (self.$y == value) {

return false;

}

self.$y = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setY(value);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

return true;

};

Object.defineProperty(DisplayObject.prototype, "scaleX", {

get: function () {

return this.$getScaleX();

},

set: function (value) {

this.$setScaleX(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getScaleX = function () {

return this.$scaleX;

};

DisplayObject.prototype.$setScaleX = function (value) {

var self = this;

self.$scaleX = value;

self.$matrixDirty = true;

self.$updateUseTransform();

if (egret.nativeRender) {

self.$nativeDisplayObject.setScaleX(value);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "scaleY", {

get: function () {

return this.$getScaleY();

},

set: function (value) {

this.$setScaleY(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getScaleY = function () {

return this.$scaleY;

};

DisplayObject.prototype.$setScaleY = function (value) {

var self = this;

self.$scaleY = value;

self.$matrixDirty = true;

self.$updateUseTransform();

if (egret.nativeRender) {

self.$nativeDisplayObject.setScaleY(value);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "rotation", {

get: function () {

return this.$getRotation();

},

set: function (value) {

this.$setRotation(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getRotation = function () {

return this.$rotation;

};

DisplayObject.prototype.$setRotation = function (value) {

value = clampRotation(value);

var self = this;

if (value == self.$rotation) {

return;

}

var delta = value - self.$rotation;

var angle = delta / 180 \* Math.PI;

self.$skewX += angle;

self.$skewY += angle;

self.$rotation = value;

self.$matrixDirty = true;

self.$updateUseTransform();

if (egret.nativeRender) {

self.$nativeDisplayObject.setRotation(value);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "skewX", {

get: function () {

return this.$skewXdeg;

},

set: function (value) {

this.$setSkewX(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setSkewX = function (value) {

var self = this;

if (value == self.$skewXdeg) {

return;

}

self.$skewXdeg = value;

value = clampRotation(value);

value = value / 180 \* Math.PI;

self.$skewX = value;

self.$matrixDirty = true;

self.$updateUseTransform();

if (egret.nativeRender) {

self.$nativeDisplayObject.setSkewX(self.$skewXdeg);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "skewY", {

get: function () {

return this.$skewYdeg;

},

set: function (value) {

this.$setSkewY(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setSkewY = function (value) {

var self = this;

if (value == self.$skewYdeg) {

return;

}

self.$skewYdeg = value;

value = clampRotation(value);

value = value / 180 \* Math.PI;

self.$skewY = value;

self.$matrixDirty = true;

self.$updateUseTransform();

if (egret.nativeRender) {

self.$nativeDisplayObject.setSkewY(self.$skewYdeg);

}

else {

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "width", {

get: function () {

return this.$getWidth();

},

set: function (value) {

this.$setWidth(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getWidth = function () {

var self = this;

return isNaN(self.$explicitWidth) ? self.$getOriginalBounds().width : self.$explicitWidth;

};

DisplayObject.prototype.$setWidth = function (value) {

this.$explicitWidth = isNaN(value) ? NaN : value;

};

Object.defineProperty(DisplayObject.prototype, "height", {

get: function () {

return this.$getHeight();

},

set: function (value) {

this.$setHeight(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getHeight = function () {

var self = this;

return isNaN(self.$explicitHeight) ? self.$getOriginalBounds().height : self.$explicitHeight;

};

DisplayObject.prototype.$setHeight = function (value) {

this.$explicitHeight = isNaN(value) ? NaN : value;

};

Object.defineProperty(DisplayObject.prototype, "measuredWidth", {

get: function () {

return this.$getOriginalBounds().width;

},

enumerable: true,

configurable: true

});

Object.defineProperty(DisplayObject.prototype, "measuredHeight", {

get: function () {

return this.$getOriginalBounds().height;

},

enumerable: true,

configurable: true

});

Object.defineProperty(DisplayObject.prototype, "anchorOffsetX", {

get: function () {

return this.$anchorOffsetX;

},

set: function (value) {

this.$setAnchorOffsetX(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setAnchorOffsetX = function (value) {

var self = this;

self.$anchorOffsetX = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setAnchorOffsetX(value);

}

};

Object.defineProperty(DisplayObject.prototype, "anchorOffsetY", {

get: function () {

return this.$anchorOffsetY;

},

set: function (value) {

this.$setAnchorOffsetY(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setAnchorOffsetY = function (value) {

var self = this;

self.$anchorOffsetY = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setAnchorOffsetY(value);

}

};

Object.defineProperty(DisplayObject.prototype, "visible", {

get: function () {

return this.$visible;

},

set: function (value) {

this.$setVisible(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setVisible = function (value) {

var self = this;

self.$visible = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setVisible(value);

}

else {

self.updateRenderMode();

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "cacheAsBitmap", {

get: function () {

return this.$cacheAsBitmap;

},

set: function (value) {

var self = this;

self.$cacheAsBitmap = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setCacheAsBitmap(value);

}

else {

self.$setHasDisplayList(value);

}

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setHasDisplayList = function (value) {

var self = this;

var hasDisplayList = !!self.$displayList;

if (hasDisplayList == value) {

return;

}

if (value) {

var displayList = egret.sys.DisplayList.create(self);

if (displayList) {

self.$displayList = displayList;

self.$cacheDirty = true;

}

}

else {

self.$displayList = null;

}

};

DisplayObject.prototype.$cacheDirtyUp = function () {

var p = this.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

};

Object.defineProperty(DisplayObject.prototype, "alpha", {

get: function () {

return this.$alpha;

},

set: function (value) {

this.$setAlpha(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setAlpha = function (value) {

var self = this;

self.$alpha = value;

if (egret.nativeRender) {

self.$nativeDisplayObject.setAlpha(value);

}

else {

self.updateRenderMode();

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

};

Object.defineProperty(DisplayObject.prototype, "touchEnabled", {

get: function () {

return this.$getTouchEnabled();

},

set: function (value) {

this.$setTouchEnabled(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$getTouchEnabled = function () {

return this.$touchEnabled;

};

DisplayObject.prototype.$setTouchEnabled = function (value) {

this.$touchEnabled = value;

};

Object.defineProperty(DisplayObject.prototype, "scrollRect", {

get: function () {

return this.$scrollRect;

},

set: function (value) {

this.$setScrollRect(value);

},

enumerable: true,

configurable: true

});

DisplayObject.prototype.$setScrollRect = function (value) {

var self = this;

if (!value && !self.$scrollRect) {

self.updateRenderMode();

return;

}

if (value) {

if (!self.$scrollRect) {

self.$scrollRect = new egret.Rectangle();

}

self.$scrollRect.copyFrom(value);

if (egret.nativeRender) {

self.$nativeDisplayObject.setScrollRect(value.x, value.y, value.width, value.height);

}

}

else {

self.$scrollRect = null;

if (egret.nativeRender) {

self.$nativeDisplayObject.setScrollRect(0, 0, 0, 0);

}

}

if (!egret.nativeRender) {

self.updateRenderMode();

}

};

Object.defineProperty(DisplayObject.prototype, "blendMode", {

get: function () {

return egret.sys.numberToBlendMode(this.$blendMode);

},

set: function (value) {

var self = this;

var mode = egret.sys.blendModeToNumber(value);

self.$blendMode = mode;

if (egret.nativeRender) {

self.$nativeDisplayObject.setBlendMode(mode);

}

else {

self.updateRenderMode();

var p = self.$parent;

if (p && !p.$cacheDirty) {

p.$cacheDirty = true;

p.$cacheDirtyUp();

}

var maskedObject = self.$maskedObject;

if (maskedObject && !maskedObject.$cacheDirty) {

maskedObject.$cacheDirty = true;

maskedObject.$cacheDirtyUp();

}

}

},

enumerable: true,

configurable: true

});

Object.defineProperty(DisplayObject.prototype, "mask", {

get: function () {

var self = this;

return self.$mask ? self.$mask : self.$maskRect;

},

set: function (value) {

var self = this;

if (value === self) {

return;

}

if (value) {

if (value instanceof DisplayObject) {

if (value == self.$mask) {

return;

}

if (value.$maskedObject) {

value.$maskedObject.mask = null;

}

value.$maskedObject = self;

self.$mask = value;

if (!egret.nativeRender) {

value.updateRenderMode();

}

if (self.$maskRect) {

if (egret.nativeRender) {

self.$nativeDisplayObject.setMaskRect(0, 0, 0, 0);

}

self.$maskRect = null;

}

if (egret.nativeRender) {

self.$nativeDisplayObject.setMask(value.$nativeDisplayObject.id);

}

}

else {

if (!self.$maskRect) {

self.$maskRect = new egret.Rectangle();

}

self.$maskRect.copyFrom(value);

if (egret.nativeRender) {

self.$nativeDisplayObject.setMaskRect(value.x, value.y, value.width, value.height);

}

if (self.$mask) {

self.$mask.$maskedObject = null;

if (!egret.nativeRender) {

self.$mask.updateRenderMode();

}

}

if (self.mask) {

if (egret.nativeRender) {

self.$nativeDisplayObject.setMask(-1);

}

self.$mask = null;

}

}

}

else {

if (self.$mask) {

self.$mask.$maskedObject = null;

if (!egret.nativeRender) {

self.$mask.updateRenderMode();

}

}

if (self.mask) {

if (egret.nativeRender) {

self.$nativeDisplayObject.setMask(-1);

}

self.$mask = null;

}

if (self.$maskRect) {

if (egret.nativeRender) {

self.$nativeDisplayObject.setMaskRect(0, 0, 0, 0);

}

self.$maskRect = null;

}

}

if (!egret.nativeRender) {

self.updateRenderMode();

}

},

enumerable: true,

configurable: true

});

var egret;

(function (egret) {

var BlurFilter = (function (\_super) {

\_\_extends(BlurFilter, \_super);

function BlurFilter(blurX, blurY, quality) {

if (blurX === void 0) { blurX = 4; }

if (blurY === void 0) { blurY = 4; }

if (quality === void 0) { quality = 1; }

var \_this = \_super.call(this) || this;

var self = \_this;

self.type = "blur";

self.$blurX = blurX;

self.$blurY = blurY;

self.$quality = quality;

self.blurXFilter = new BlurXFilter(blurX);

self.blurYFilter = new BlurYFilter(blurY);

self.onPropertyChange();

return \_this;

}

Object.defineProperty(BlurFilter.prototype, "blurX", {

get: function () {

return this.$blurX;

},

set: function (value) {

var self = this;

if (self.$blurX == value) {

return;

}

self.$blurX = value;

self.blurXFilter.blurX = value;

self.onPropertyChange();

},

enumerable: true,

configurable: true

});

Object.defineProperty(BlurFilter.prototype, "blurY", {

get: function () {

return this.$blurY;

},

set: function (value) {

var self = this;

if (self.$blurY == value) {

return;

}

self.$blurY = value;

self.blurYFilter.blurY = value;

self.onPropertyChange();

},

enumerable: true,

configurable: true

});

}

class GameEvent {

public constructor() {

}

}

class GameUI extends eui.Component {

public constructor() {

super();

this.skinName = "GameSkin";

}

private gp\_market: eui.Group;

private gp\_store: eui.Group;

private gp\_over: eui.Group;

private lbl\_day: eui.Label;

private lbl\_store: eui.Label;

private lbl\_1: eui.Label;

private lbl\_2: eui.Label;

private lbl\_3: eui.Label;

private lbl\_4: eui.Label;

private lbl\_5: eui.Label;

private lbl\_6: eui.Label;

private rect\_evt: eui.Rect;

private cb\_0:eui.CheckBox;

private crtPop: number;

private market\_arr: MarketItem[];

private store\_arr: StoreItem[];

private gamestate: number;

private max\_num: number;

private data: msgLifeDataRsp;

private leftStore: number;

protected childrenCreated() {

super.childrenCreated();

GameLogic.getInstance().gameui = this;

this.market\_arr = [];

this.store\_arr = [];

this.eventlist = [];

this.initView();

this.initEvent();

GameCommand.getInstance().startGame();

}

private eventlist: string[];

private eventpoping: boolean;

public eventAppear(str: string) {

if(GameLogic.getInstance().cbSelected){

return;

}

if (this.eventpoping) {

this.eventlist.push(str);

return;

}

this.popEvent(str);

}

private eventNext() {

this.eventpoping = false;

if (this.eventlist.length > 0) {

let str = this.eventlist.shift();

this.popEvent(str);

}

else {

this.pop(0);

}

}

private popEvent(str: string) {

this.eventpoping = true;

this.pop(11);

this['lbl\_event\_1'].text = str;

}

public initData(msg: msgLifeDataRsp) {

this.data = msg;

this.setLeft();

this.lbl\_1.text = this.data.dwMoney.toString();

this.lbl\_2.text = this.data.dwDeposit.toString();

this.lbl\_3.text = this.data.dwDebt.toString();

this.lbl\_4.text = this.data.dwPow.toString();

this.lbl\_5.text = this.data.dwFame.toString();

let maxday = GameLogic.getInstance().data['maxday'];

this.lbl\_day.text = (maxday - this.data.dwTimes) + "/" + maxday + "天";

}

private setLeft() {

let n = this.getStoreNum();

this.leftStore = this.data.dwMaxStoreNum - n;

this.lbl\_store.text = n + "/" + this.data.dwMaxStoreNum;

}

private getStoreNum(): number {

let n: number = 0;

for (let i: number = 0; i < this.store\_arr.length; i++) {

let item = this.store\_arr[i];

n += item.good.dwNum;

}

return n;

}

public initMarket(msg: msgGoodsBuyRsp) {

this.clearMarket();

for (let i: number = 0; i < msg.goods.length; i++) {

let item = new MarketItem(msg.goods[i]);

item.y = (item.height + 2) \* i;

item.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickMarketItem, this);

this.market\_arr.push(item);

this.gp\_market.addChild(item);

}

}

private crtMarketItem: MarketItem;

private clickMarketItem(e: egret.TouchEvent) {

let item = e.currentTarget as MarketItem;

if (this.crtMarketItem != null) {

this.crtMarketItem.select = false;

}

this.crtMarketItem = item;

this.crtMarketItem.select = true;

this.pop(9);

let max = Math.floor(this.data.dwMoney / item.good.dwPrice);

this.max\_num = max > this.leftStore ? this.leftStore : max;

this['lbl\_buy\_1'].text = item.good.strName;

this['lbl\_num6'].text = this.max\_num + "";

}

public initStore(msg: msgGoodsStoreRsp): void {

this.clearStore();

for (let i: number = 0; i < msg.goods.length; i++) {

let item = new StoreItem(msg.goods[i]);

item.y = (item.height + 2) \* i;

item.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickStoreItem, this);

this.store\_arr.push(item);

this.gp\_store.addChild(item);

}

this.setLeft();

}

private crtStoreItem: StoreItem;

private clickStoreItem(e: egret.TouchEvent) {

let item = e.currentTarget as StoreItem;

if (this.crtStoreItem != null) {

this.crtStoreItem.select = false;

}

this.crtStoreItem = item;

this.crtStoreItem.select = true;

this.pop(10);

this.max\_num = this.crtStoreItem.good.dwNum;

this['lbl\_sell\_1'].text = item.good.strName;

this['lbl\_num7'].text = this.max\_num + "";

}

public storeUp(arr): void {

}

public over() {

this['gp\_over'].visible = true;

let str:string = "";

if (this.data.dwPow <= 0) {

str += StringUtil.getSwfLangStr("s20") + "\n";

this['btn\_27'].visible = false;

}

else {

str += StringUtil.getSwfLangStr("s11") + "\n";

str += StringUtil.getSwfLangStrVarByID("s21", [DataBase.money + ""]) + "\n";

str += StringUtil.getSwfLangStr("s12") + "\n";

for (let i: number = 0; i < 5; i++) {

str += StringUtil.getSwfLangStrVarByID("s1" + (3 + i), [DataBase.achives[i] + ""]) + "\n";

}

str += StringUtil.getSwfLangStr("s19") + "\n";

str += StringUtil.getSwfLangStr("s50") + "\n";

this['btn\_27'].visible = true;

}

this['lbl\_over\_1'].text = str;

}

public errorRsp(i: number) {

this.eventAppear(StringUtil.getSwfLangStr("e" + i));

}

private initView() {

this.cb\_0.selected = GameLogic.getInstance().cbSelected;

}

private initEvent() {

this.addEventListener(egret.Event.REMOVED\_FROM\_STAGE, this.clear, this);

for (let i: number = 0; i <= 27; i++) {

let btn: eui.Button = this['btn\_' + i];

btn.name = i + "";

btn.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickBtn, this);

}

for (let i: number = 0; i <= 7; i++) {

let lbl: eui.Label = this['lbl\_num' + i];

lbl.name = 'lbl' + i;

lbl.addEventListener(egret.Event.CHANGE, this.txtChange, this);

lbl.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.txtClick, this);

}

this.cb\_0.addEventListener(egret.Event.CHANGE,this.cbChange,this);

this.rect\_evt.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickRect, this);

}

private cbChange(){

GameLogic.getInstance().cbSelected = this.cb\_0.selected;

}

private clickRect(e: egret.TouchEvent) {

this.eventNext();

}

private txtClick(e: egret.TouchEvent) {

let lbl: eui.Label = e.currentTarget as eui.Label;

let i = parseInt(lbl.name.slice(3));

switch (i) {

case 1://存款

this.max\_num = this.data.dwMoney;

break;

case 2://取款

this.max\_num = this.data.dwDeposit;

break;

}

}

private txtChange(e: egret.Event) {

let lbl: eui.Label = e.currentTarget as eui.Label;

let n = parseInt(lbl.text);

if (n > this.max\_num) {

lbl.text = this.max\_num + "";

}

}

private clickBtn(e: egret.TouchEvent) {

let i = parseInt(e.currentTarget.name);

if (this.gamestate == 0) {

if (i < 9 && i > 12 && i < 26) {

return;

}

}

switch (i) {

case 0://捐款

this.pop(0);

GameCommand.getInstance().charity(parseInt(this['lbl\_num0'].text));

break;

case 1:

case 2:

case 3:

this.pop(0);

GameCommand.getInstance().passOneDay();

break;

case 4://慈善

this.pop(i);

this.max\_num = this.data.dwMoney;

this['lbl\_charity'].text = StringUtil.getSwfLangStr("s10");

this['lbl\_num0'].text = "0";

break;

case 5://银行

this.pop(i);

this['lbl\_num1'].text = this.data.dwMoney + "";

this['lbl\_num2'].text = this.data.dwDeposit + "";

break;

case 6://医院

this.pop(i);

let n = 100 - this.data.dwPow;

if (n <= 0) {

this['lbl\_hos\_1'].text = StringUtil.getSwfLangStr("s6");

this.max\_num = 0;

}

else {

this['lbl\_hos\_1'].text = StringUtil.getSwfLangStrVarByID("s7", [GameLogic.getInstance().data['hospital'] + ""]);

this.max\_num = n;

}

this['lbl\_num3'].text = this.max\_num + "";

break;

case 7://中介

this.pop(i);

let n7 = GameLogic.getInstance().data['maxstore'] - this.data.dwMaxStoreNum;

if (n7 <= 0) {

this['lbl\_medi\_1'].text = StringUtil.getSwfLangStr("s8");

}

else {

let n70 = GameLogic.getInstance().data['storeprice'];

let n71 = Math.floor(Math.random() \* n70 / 5) + n70;

this['lbl\_medi\_1'].text = StringUtil.getSwfLangStrVarByID("s9", [n71 + ""]);

}

break;

case 8://邮局

this.pop(i);

let n8 = this.data.dwDebt;

let n80 = this.data.dwMoney > n8 ? n8 : this.data.dwMoney;

if (n8 <= 0) {//没有债务

this['lbl\_post\_1'].text = StringUtil.getSwfLangStr("s10");

this.max\_num = 0;

}

else {

this['lbl\_post\_1'].text = StringUtil.getSwfLangStr("s11");

this.max\_num = n80;

}

this['lbl\_num5'].text = this.max\_num + "";

break;

case 9://转发

GameLogic.getInstance().share(1);

break;

case 10://广告

break;

case 11://排行榜

break;

case 12://重新开始

case 26:

this.restart();

break;

case 13://存钱

GameCommand.getInstance().cun(parseInt(this['lbl\_num1'].text));

this.pop(0);

break;

case 14://取钱

GameCommand.getInstance().qu(parseInt(this['lbl\_num2'].text));

this.pop(0);

break;

case 15://治疗

if (this.data.dwPow < 100) {

GameCommand.getInstance().treat(100 - this.data.dwPow);

}

this.pop(0);

break;

case 17://买柜子

let n17 = GameLogic.getInstance().data['storeprice'];

n17 = Math.floor(Math.random() \* n17 / 5) + n17;

GameCommand.getInstance().buyStore(n17);

this.pop(0);

break;

case 19://还债

let n19 = parseInt(this['lbl\_num5'].text);

n19 = this.data.dwMoney < n19 ? this.data.dwMoney : n19;

GameCommand.getInstance().huan(n19);

this.pop(0);

break;

case 21://购买

let n21 = parseInt(this['lbl\_num6'].text);

if (n21 > 0) {

GameCommand.getInstance().buyGoods(this.crtMarketItem.good.dwID, n21);

}

this.pop(0);

break;

case 23://出售

let n23 = parseInt(this['lbl\_num7'].text);

if (n23 > 0) {

GameCommand.getInstance().sellGoods(this.crtStoreItem.good.dwID, n23);

}

this.pop(0);

break;

case 16://关闭

case 18:

case 20:

case 22:

case 24:

this.pop(0);

break;

case 25:

this.eventNext();

break;

case 27://炫耀

break;

}

}

private pop(i: number) {

if (this.crtPop != null) {

let gp = this['gp\_' + this.crtPop];

if (gp != null) {

gp.visible = false;

}

}

if (i == 0) {

this.eventpoping = false;

}

let gp = this['gp\_' + i];

if (gp != null) {

gp.visible = true;

this.crtPop = i;

}

else {

this.crtPop = null;

}

}

private clearMarket() {

for (let i: number = 0; i < this.market\_arr.length; i++) {

let item = this.market\_arr[i];

}

this.gp\_market.removeChildren();

this.market\_arr = [];

this.crtMarketItem = null;

}

private clearStore() {

for (let i: number = 0; i < this.store\_arr.length; i++) {

let item = this.store\_arr[i];

item.removeEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickStoreItem, this);

}

this.gp\_store.removeChildren();

this.store\_arr = [];

this.crtStoreItem = null;

}

private restart() {

GameLogic.getInstance().openStart();

}

private clear() {

this.clearEvent();

this.clearMarket();

this.clearStore();

GameLogic.getInstance().gameui = null;

}

private clearEvent() {

this.removeEventListener(egret.Event.REMOVED\_FROM\_STAGE, this.clear, this);

for (let i: number = 0; i <= 27; i++) {

let btn: eui.Button = this['btn\_' + i];

btn.removeEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickBtn, this);

}

for (let i: number = 0; i <= 7; i++) {

let lbl: eui.Label = this['lbl\_num' + i];

lbl.removeEventListener(egret.Event.CHANGE, this.txtChange, this);

lbl.removeEventListener(egret.TouchEvent.TOUCH\_TAP, this.txtClick, this);

}

this.cb\_0.removeEventListener(egret.Event.CHANGE,this.cbChange,this);

this.rect\_evt.removeEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickRect, this);

}

}

class StoreItem extends eui.Component {

public constructor(o:varGoods) {

super();

this.good = o;

this.skinName = "StoreItemSkin";

}

public lbl\_name:eui.Label;

public lbl\_price:eui.Label;

public lbl\_num:eui.Label;

public rect\_bg:eui.Rect;

public img\_up:eui.Image;

public good:varGoods;

protected childrenCreated(){

super.childrenCreated();

this.update();

}

public updateGood(o:varGoods){

this.good = o;

this.update();

}

private update(){

this.lbl\_name.text = this.good.strName;

this.lbl\_price.text = this.good.dwPrice + "";

this.lbl\_num.text = this.good.dwNum + "";

}

private \_select:boolean;

public set select(b:boolean){

this.\_select = b;

this.rect\_bg.fillAlpha = b ? 1 : 0;

}

public get select():boolean{

return this.\_select;

}

public clear(){

this.good = null;

}

}

class StartUI extends eui.Component {

public constructor() {

super();

this.skinName = "StartSkin";

}

private lbl\_name: eui.Label;

private lbl\_content: eui.Label;

protected childrenCreated() {

super.childrenCreated();

let data = GameLogic.getInstance().data;

this.lbl\_name.text = StringUtil.getSwfLangStr("s1");

this.lbl\_content.text = StringUtil.getSwfLangStr("s2");

for (let i: number = 1; i <= 3; i++) {

let o = data['config' + i];

let str = i == 1 ? "开始游戏" : o['rmb'] + "元 开始";

str += "\n" + o['pow'] + "体力 " + o['debt'] + "欠债 " + o['money'] + "启动资金\n";

if (i == 3) {

str += "(可获炒房证)";

}

let btn:eui.Button = this['btn\_' + i];

btn.label = str;

btn.name = i + "";

btn.addEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickBtn, this);

}

this.addEventListener(egret.Event.REMOVED\_FROM\_STAGE,this.clear,this);

}

private clickBtn(e: egret.TouchEvent) {

let i = parseInt(e.currentTarget.name);

GameCommand.getInstance().selectPackage(i);

GameLogic.getInstance().startGame();

}

private clear() {

for (let i: number = 1; i <= 3; i++) {

this['btn\_' + i].removeEventListener(egret.TouchEvent.TOUCH\_TAP, this.clickBtn, this);

}

this.removeEventListener(egret.Event.REMOVED\_FROM\_STAGE,this.clear,this);

}

}

class MarketItem extends eui.Component {

public constructor(o:varGoods) {

super();

this.good = o;

this.skinName = "MarketItemSkin";

}

public lbl\_name:eui.Label;

public lbl\_price:eui.Label;

public rect\_bg:eui.Rect;

public good:varGoods;

protected childrenCreated(){

super.childrenCreated();

this.lbl\_name.text = this.good.strName;

this.lbl\_price.text = this.good.dwPrice + "";

}

private \_select:boolean;

public set select(b:boolean){

this.\_select = b;

this.rect\_bg.fillAlpha = b ? 1 : 0;

}

public get select():boolean{

return this.\_select;

}

public clear(){

this.good = null;

}

}