

```
1 class Main extends eui.UILayer {
2     protected createChildren(): void {
3         super.createChildren();
4         GameLogic.getInstance().GameStage = this.stage;
5         GameLogic.getInstance().main = this;
6         egret.lifecycle.addLifecycleListener((context) => {
7             // custom lifecycle plugin
8         })
9         egret.lifecycle.onPause = () => {
10             egret.ticker.pause();
11         }
12         egret.lifecycle.onResume = () => {
13             egret.ticker.resume();
14         }
15         let assetAdapter = new AssetAdapter();
16         egret.registerImplementation("eui.IAssetAdapter", assetAdapter);
17         egret.registerImplementation("eui.IThemeAdapter", new ThemeAdapter());
18         this.runGame().catch(e => {
19             console.log(e);
20         })
21     }
22     private async runGame() {
23         await this.loadResource()
24         this.createGameScene();
25         await platform.login();
26         const userInfo = await platform.getUserInfo();
27     }
28 }
29 private async loadResource() {
30     try {
31         const loadingView = new LoadingUI();
32         this.stage.addChild(loadingView);
33         await RES.loadConfig("resource/default.res.json", "resource/");
34         await this.loadTheme();
35         await RES.loadGroup("preload", 0, loadingView);
36         this.stage.removeChild(loadingView);
37     }
38     catch (e) {
39         console.error(e);
40     }
41 }
42 private loadTheme() {
43     return new Promise((resolve, reject) => {
44         // load skin theme configuration file, you can manually modify the file. And
45         replace the default skin.
46         let theme = new eui.Theme("resource/default.thm.json", this.stage);
47         theme.addEventListener(eui.UIEvent.COMPLETE, () => {
48             resolve();
49         }, this);
50     })
}
```

```
1      }
2      protected createGameScene(): void {
3          GameLogic.getInstance().openStart();
4      }
5  }
6  class LoadingUI extends egret.Sprite implements RES.PromiseTaskReporter {
7      public constructor() {
8          super();
9          this.createView();
10     }
11     private textField: egret.TextField;
12     private createView(): void {
13         this.textField = new egret.TextField();
14         this.addChild(this.textField);
15         this.textField.y = 300;
16         this.textField.width = 480;
17         this.textField.height = 100;
18         this.textField.textAlign = "center";
19     }
20     public onProgress(current: number, total: number): void {
21         this.textField.text = `Loading...${current}/${total}`;
22     }
23 }
24 class AssetAdapter implements eui.IAssetAdapter {
25     public getAsset(source: string, compFunc:Function, thisObject: any): void {
26         function onGetRes(data: any): void {
27             compFunc.call(thisObject, data, source);
28         }
29         if (RES.hasRes(source)) {
30             let data = RES.getRes(source);
31             if (data) {
32                 onGetRes(data);
33             }
34             else {
35                 RES.getResAsync(source, onGetRes, this);
36             }
37         }
38         else {
39             RES.getResByUrl(source, onGetRes, this, RES.ResourceItem.TYPE_IMAGE);
40         }
41     }
42 }
43 declare interface Platform {
44     getUserInfo(): Promise<any>;
45     login(): Promise<any>
46 }
47
48 class DebugPlatform implements Platform {
49     async getUserInfo() {
50         return { nickName: "username" }
```

```
1      }
2      async login() {
3      }
4  }
5  if (!window.platform) {
6      window.platform = new DebugPlatform();
7  }
8  declare let platform: Platform;
9  declare interface Window {
10      platform: Platform
11  }
12  class ThemeAdapter implements eui.IThemeAdapter {
13      public getTheme(url: string, onSuccess: Function, onError: Function, thisObject: any): void {
14          function onResGet(e: string): void {
15              onSuccess.call(thisObject, e);
16          }
17          function onResError(e: RES.ResourceEvent): void {
18              if (e.resItem.url == url) {
19                  RES.removeEventListener(RES.ResourceEvent.ITEM_LOAD_ERROR,
20 onResError, null);
21                  onError.call(thisObject);
22              }
23          }
24          if (typeof generateEUI !== 'undefined') {
25              egret.callLater(() => {
26                  onSuccess.call(thisObject, generateEUI);
27              }, this);
28          }
29          else {
30              RES.addEventListener(RES.ResourceEvent.ITEM_LOAD_ERROR, onResError, null);
31              RES.getResByUrl(url, onResGet, this, RES.ResourceItem.TYPE_TEXT);
32          }
33      }
34  }
35  declare var generateEUI: { paths: string[], skins: any }
36  class GameCommand extends egret.EventDispatcher {
37      public constructor() {
38          super();
39      }
40      private static _instance: GameCommand;
41      public static getInstance(): GameCommand {
42          if (this._instance == null) {
43              this._instance = new GameCommand();
44          }
45          return this._instance;
46      }
47      public sendData(b: boolean = false) {
48          if (b) {
49              DataBase.debt = Math.floor(DataBase.debt * 1.15);
50              DataBase.deposit = Math.floor(DataBase.deposit * 1.04);
```

```
1      }
2      let msg = this.getData();
3      GameLogic.getInstance().gameui.initData(msg);
4  }
5  public sendMarket(evt: boolean) {
6      DataBase.events = [];
7      DataBase.marketGoods = [];
8      let msg: msgGoodsBuyRsp = this.getMarket(evt);
9      DataBase.marketGoods = msg.goods;
10     GameLogic.getInstance().gameui.initMarket(msg);
11 }
12 public sendStore() {
13     let msg = new msgGoodsStoreRsp();
14     msg.goods = DataBase.storeGoods;
15     GameLogic.getInstance().gameui.initStore(msg);
16 }
17 public sendEvent() {
18     this.dealOtherEvent();
19     let arr = DataBase.events;
20     for (let i: number = 0; i < arr.length; i++) {
21         GameLogic.getInstance().gameui.eventAppear(arr[i]);
22     }
23     DataBase.events = [];
24 }
25 public sendError(i: number) {
26     GameLogic.getInstance().gameui.errorRsp(i);
27 }
28 public sendOver(t:number) {
29     DataBase.gameState = 0;
30     if(t == 0){//时间到
31         DataBase.debt = Math.floor(DataBase.debt * 1.15);
32         DataBase.deposit = Math.floor(DataBase.deposit * 1.04);
33         DataBase.money = DataBase.money + DataBase.deposit - DataBase.debt +
34 this.getStorePrice();
35         DataBase.debt = 0;
36         DataBase.deposit = 0;
37     }
38     else if(t == 1){//体力为 0
39     }
40     this.sendData();
41     GameLogic.getInstance().gameui.over();
42 }
43 private getStorePrice():number{
44     let p:number = 0;
45     for(let i:number=0;i<DataBase.storeGoods.length;i++){
46         let good = DataBase.storeGoods[i];
47         p += good.dwPrice * good.dwNum;
48     }
49     return p;
50 }
```

```
1      public getData(): msgLifeDataRsp {
2          let msg = new msgLifeDataRsp();
3          msg.dwMoney = DataBase.money;
4          msg.dwDebt = DataBase.debt;
5          msg.dwDeposit = DataBase.deposit;
6          msg.dwPow = DataBase.pow;
7          msg.dwTimes = DataBase.times;
8          msg.dwMaxStoreNum = DataBase.maxStoreNum;
9          msg.dwFame = DataBase.fame;
10         return msg;
11     }
12     private bases: number[] = [1, 2, 3, 4, 5, 6, 7, 8, 9];
13     public getMarket(evt: boolean): msgGoodsBuyRsp {
14         let msg = new msgGoodsBuyRsp();
15         msg.goods = [];
16         let len = 4 + Math.floor(Math.random() * 6);
17         let arr = this.bases.slice();
18         let lll = DataBase.gamePackage < 2 ? arr.length : arr.length - 1;
19         let goodIds = [];
20         for (let i: number = 0; i < len; i++) {
21             let i = Math.floor(Math.random() * lll);
22             goodIds.push(arr[i]);
23             arr.splice(i, 1);
24         }
25         goodIds.sort(this.sortfun);
26         for (let i: number = 0; i < goodIds.length; i++) {
27             let good = new varGoods();
28             let id = goodIds[i];
29             let o = GameLogic.getInstance().goods[id];
30             if (o == null) {
31                 continue;
32             }
33             good.dwID = id;
34             good.strName = o['name'];
35             good.dwPrice = this.getPrice(o, evt);
36             good.dwNum = 0;
37             msg.goods.push(good);
38         }
39         return msg;
40     }
41     private dealOtherEvent() {
42         let b = Math.random() < 0.2;
43         if (b) {
44             let a = Math.floor(Math.random() * 4) + 1;
45             let b = Math.random() < 0.5 ? 1 : 2;
46             let c = Math.floor(Math.random() * 3) + 1;
47             this.addEvent(a, b, c);
48         }
49     }
50     private addEvent(a, b, c) {
```

```
1      let id = a * 100 + b * 10 + c;
2      let o = GameLogic.getInstance().goods["evt" + id];
3      if (o == null) {
4          return;
5      }
6      let isadd = b == 2;
7      let value = o['value'];
8      if (a < 5) {
9          switch (a) {
10             case 1://money
11                 if (value <= 1) {
12                     value = Math.floor(DataBase.money * value);
13                 }
14                 else {
15                     value = Math.floor(Math.random() * value / 5);
16                 }
17                 DataBase.money = DataBase.money + (isadd ? value : -value);
18                 DataBase.money = DataBase.money <= 0 ? 0 : DataBase.money;
19                 break;
20             case 2://deposit
21                 if (value <= 1) {
22                     value = Math.floor(DataBase.money * value);
23                 }
24                 else {
25                     value = Math.floor(Math.random() * value / 5);
26                 }
27                 DataBase.deposit = DataBase.deposit + (isadd ? value : -value);
28                 DataBase.deposit = DataBase.deposit <= 0 ? 0 : DataBase.deposit;
29                 break;
30             case 3://pow
31                 DataBase.pow = DataBase.pow + (isadd ? value : -value);
32                 DataBase.pow = DataBase.pow <= 0 ? 0 : DataBase.pow;
33                 break;
34             case 4://fame
35                 DataBase.fame = DataBase.fame + (isadd ? value : -value);
36                 DataBase.fame = DataBase.fame <= 0 ? 0 : DataBase.fame;
37                 break;
38             }
39         }
40         if(typeof(o) == "string"){//其他事件
41             DataBase.events.push(StringUtil.getSwfLangStr(o));
42         }
43         else{//商品事件
44             DataBase.events.push(StringUtil.getSwfLangStrVar(o['str'], [value]));
45         }
46     }
47     private diss1: number[] = [0.2, 5, 10];
48     private getRandom1(): number {
49         let r = Math.random();
50         if (r < 0.1) {
```

```
1         let i = Math.floor(Math.random() * 3);
2         return this.diss1[i];
3     }
4     else {
5         return 1;
6     }
7 }
8 private diss2: number[] = [0.1, 0.2, 5, 10];
9 private getRandom2(): number {
10     let i = Math.floor(Math.random() * 4);
11     return this.diss2[i];
12 }
13 private getPrice(o: Object, evt: boolean): number {
14     let n = o['price'];
15     let r1 = this.getRandom1();
16     let v = Math.floor(n * r1);
17     let b2 = Math.random() < 0.5;
18     let v2 = Math.floor(v * Math.random() * 0.2);
19     v = b2 ? v + v2 : v - v2;
20     if (evt) {
21         let b3 = Math.random() < 0.1;
22         if (b3) {
23             let r3 = this.getRandom2();
24             v = Math.floor(v * r3);
25             let r4 = Math.floor(Math.random() * 3) + 1;
26             let evt = 'evt' + (r3 < 1 ? 0 : 1) + r4;
27             DataBase.events.push(o[evt]);
28         }
29     }
30     return v;
31 }
32 private sortfun(a: number, b: number): number {
33     return a < b ? -1 : 1;
34 }
35 private saveAchieve(){
36     if(DataBase.money > DataBase.achives[0]){
37         DataBase.achives[0] = DataBase.money;
38     }
39     if(DataBase.deposit > DataBase.achives[1]){
40         DataBase.achives[1] = DataBase.deposit;
41     }
42     if(DataBase.debt > DataBase.achives[2]){
43         DataBase.achives[2] = DataBase.debt;
44     }
45     if(DataBase.pow < DataBase.achives[3]){
46         DataBase.achives[3] = DataBase.pow;
47     }
48     if(DataBase.fame < DataBase.achives[4]){
49         DataBase.achives[4] = DataBase.fame;
50     }
```

```
1         if(DataBase.fame > DataBase.achives[5]){
2             DataBase.achives[5] = DataBase.fame;
3         }
4     }
5     private getPriceInMarket(id: number): number {
6         let arr = DataBase.marketGoods;
7         for (let i: number = 0; i < arr.length; i++) {
8             let good = arr[i];
9             if (id == good.dwID) {
10                 return good.dwPrice;
11             }
12         }
13         return null;
14     }
15     public selectPackage(i: number) {
16         DataBase.gamePackage = i;
17     }
18     public startGame() {
19         let o = GameLogic.getInstance().data["config" + DataBase.gamePackage];
20         DataBase.times = 1;
21         DataBase.money = o['money'];//new Int64(o['money'], 0);
22         DataBase.debt = o['debt'];
23         DataBase.deposit = 0;//new Int64(0, 0);
24         DataBase.pow = o['pow'];
25         DataBase.maxStoreNum = 100;
26         DataBase.fame = o['fame'];
27         DataBase.marketGoods = [];
28         DataBase.storeGoods = [];
29         DataBase.events = [];
30         DataBase.achives = [0,0,0,0,0];
31         DataBase.gameState = 1;
32         this.sendData();
33         this.sendMarket(false);
34     }
35     public passOneDay() {
36         if (DataBase.gameState == 0) {
37             return;
38         }
39         DataBase.times++;
40         if (DataBase.times >= 40) {
41             this.sendOver(0);
42             return;
43         }
44         this.sendMarket(true);
45         this.sendEvent();
46         this.sendData(true);
47         if (DataBase.pow <= 0) {
48             this.sendOver(1);
49             return;
50         }
```



```
1      this.saveAchieve();
2  }
3  public buyGoods(id: number, num: number) {
4      if (num == 0) {
5          this.sendError(ERROR.BUY_ZERO);
6          return;
7      }
8      if(id == 9 && DataBase.gamePackage != 3){
9          this.sendError(ERROR.NEED_LICIENCE);
10         return;
11     }
12     let arr = DataBase.marketGoods;
13     for (let i: number = 0; i < arr.length; i++) {
14         let good = arr[i];
15         if (good.dwID == id) {
16             let n = good.dwPrice * num;
17             if (n > DataBase.money) {
18                 this.sendError(ERROR.MONEY_NOT_ENOUGH);
19                 return;
20             }
21             else {
22                 let arr1 = DataBase.storeGoods;
23                 let total = 0;
24                 let index;
25                 for (let j: number = 0; j < arr1.length; j++) {
26                     let good1 = arr1[j];
27                     if (good1.dwID == id) {
28                         index = j;
29                     }
30                     total += good1.dwNum;
31                 }
32                 if (total + num > DataBase.maxStoreNum) { //柜子不够
33                     this.sendError(ERROR.STORE_NOT_ENOUGH);
34                     return;
35                 }
36                 else {
37                     DataBase.money -= n;
38                     let g = arr1[index];
39                     if (g == null) {
40                         g = new varGoods();
41                         g.dwID = id;
42                         g.dwPrice = good.dwPrice;
43                         g.dwNum = num;
44                         g.strName = good.strName;
45                         arr1.push(g);
46                     }
47                     else {
48                         let nn = g.dwNum + num;
49                         let p = Math.floor((g.dwPrice * g.dwNum + good.dwPrice *
50 num) / nn);
```

```
1             g.dwNum = nn;
2             g.dwPrice = p;
3             arr1[index] = g;
4         }
5
6         this.sendData();
7         this.sendStore();
8     }
9 }
10 break;
11 }
12 }
13 }
14 public sellGoods(id: number, num: number) {
15     if (num == 0) {
16         this.sendError(ERROR.SELL_ZERO);
17         return;
18     }
19     let marketprice = this.getPriceInMarket(id);
20     if (marketprice == null) {
21         this.sendError(ERROR.MARKET_NO_GOOD);
22         return;
23     }
24     let arr = DataBase.storeGoods;
25     for (let i: number = 0; i < arr.length; i++) {
26         let good = arr[i];
27         if (good.dwID == id) {
28             DataBase.money += marketprice * num;
29             good.dwNum -= num;
30             if (good.dwNum <= 0) {
31                 DataBase.storeGoods.splice(i, 1);
32             }
33             this.sendData();
34             this.sendStore();
35             break;
36         }
37     }
38 }
39 public cun(num: number) {
40     if (num > 0 && num <= DataBase.money) {
41         DataBase.deposit += num;
42         DataBase.money -= num;
43         this.sendData();
44     }
45 }
46 public qu(num: number) {
47     if (num > 0 && num <= DataBase.deposit) {
48         DataBase.deposit -= num;
49         DataBase.money += num;
50         this.sendData();
```

```
1      }
2  }
3  public huan(num: number) {
4      if (num > 0 && num <= DataBase.money && num <= DataBase.debt) {
5          DataBase.debt -= num;
6          DataBase.money -= num;
7          this.sendData();
8      }
9  }
10 public treat(n: number) {
11     if (n > 0 && n < 100) {
12         if (n + DataBase.pow > 100) {
13             n = 100 - DataBase.pow;
14         }
15         let needmoney = n * GameLogic.getInstance().data['hospital'];
16         if (needmoney >= DataBase.money) {
17             this.sendError(ERROR.MONEY_NOT_ENOUGH);
18             return;
19         }
20         DataBase.money -= needmoney;
21         DataBase.pow += n;
22         this.sendData();
23     }
24 }
25 public charity(n:number){
26     if(n > DataBase.money){
27         this.sendError(ERROR.MONEY_NOT_ENOUGH);
28         return;
29     }
30     let charity = GameLogic.getInstance().data['charity'];
31     let c:number;
32     if(n < charity){
33
34         let r = Math.random() * 100;
35         if(r < 2){
36             DataBase.fame += 3;
37             c = 0;
38         }
39         else{
40             c = 1;
41         }
42     }
43     else{
44         let i = Math.floor(n / charity);
45         DataBase.fame += i;
46         c = i < 10 ? 2 : (i < 100 ? 3 : 4);
47     }
48     this.addEvent(5,1,c);
49     DataBase.money -= n;
50     this.sendData();
```

```
1      this.sendEvent();
2  }
3  public buyStore(price: number) {
4      let max = GameLogic.getInstance().data['maxstore'];
5      if (DataBase.maxStoreNum >= max) {
6          this.sendError(ERROR.MAX_STORE_NUM);
7          return;
8      }
9      let n = GameLogic.getInstance().data['storeprice'];
10     if (price < n) {
11         console.log("价格低于标准值，请勿作弊");
12
13         return;
14     }
15     if (DataBase.money < price) {
16         this.sendError(ERROR.MONEY_NOT_ENOUGH);
17         return;
18     }
19     else {
20         DataBase.maxStoreNum += 10;
21         if (DataBase.maxStoreNum >= max) {
22             DataBase.maxStoreNum = max;
23         }
24         let r = Math.floor(Math.random() * n / 5);
25         DataBase.money -= (price + r);
26         this.sendData();
27         this.addEvent(5, 0, 0);
28         this.sendEvent();
29     }
30 }
31 }
32 declare const wx: any
33 class GameLogic extends egret.EventDispatcher {
34     public constructor() {
35         super();
36     }
37     private static _instance: GameLogic;
38     public static getInstance(): GameLogic {
39         if (this._instance == null) {
40             this._instance = new GameLogic();
41         }
42         return this._instance;
43     }
44     public GameStage: egret.Stage;
45     public main: eui.UILayer;
46     public data: Object;
47     public goods: Object;
48     public strings: Object;
49     public cbSelected: boolean;
50     public openStart() {
```

```
1      this.initData();
2      this.main.removeChildren();
3      this.main.addChild(new StartUI());
4  }
5  private initData() {
6      if (this.data == null) {
7          this.data = RES.getRes("config_json");
8      }
9      if (this.goods == null) {
10         this.goods = RES.getRes("goods_json");
11     }
12     if (this.strings == null) {
13         this.strings = RES.getRes("string_json");
14     }
15 }
16 public gameui: GameUI;
17 public startGame() {
18     this.main.removeChildren();
19     this.main.addChild(new GameUI());
20 }
21 public share(type: number) {
22     let wx = window["wx"];
23     if (wx != null) {
24         wx.onShareAppMessage(function () {
25             return {
26                 title: '转发标题',
27                 imageUrl: 'qua_3_png',
28                 success: function (res) {
29                     console.log('转发成功')
30                 }
31             }
32         });
33     }
34 }
35 }
36 class varGoods {
37     public constructor() {
38     }
39     public dwID:number = 0;
40     public strName:string;
41     public dwPrice:number = 0;
42     public dwNum:number = 0;
43 }
44 class msgLifeDataRsp {
45     public constructor() {
46     }
47     public dwPow:number = 0;
48     public dwMoney:number;//Int64;
49     public dwDebt:number = 0;
50     public dwDeposit:number;//Int64;
```

```
1      public dwTimes:number = 0;
2      public dwMaxStoreNum:number = 0;
3      public dwFame:number = 0;
4  }
5  class msgGoodsStoreRsp {
6      public constructor() {
7      }
8      public goods:varGoods[] = [];
9  }
10 class msgGoodsBuyRsp {
11     public constructor() {
12     }
13     public goods:varGoods[] = [];
14 }
15 class Int64 {
16     public constructor(lowerUint: number = 0, higherUint: number = 0) {
17         this._lowValue = lowerUint;
18         this._highValue = higherUint;
19     }
20     _highValue: number = 0;
21     public get higherUint(): number {
22         return this._highValue;
23     }
24     public set higherUint(value: number) {
25         if (this._highValue == value)
26             return;
27         this._highValue = value;
28         this.cacheBytes = null;
29         this.cacheString = [];
30     }
31     private _lowValue: number = 0;
32     public get lowerUint(): number {
33         return this._lowValue;
34     }
35     public set lowerUint(value: number) {
36         this._lowValue = value;
37         if (this._lowValue == value)
38             return;
39         this.cacheBytes = null;
40         this.cacheString = [];
41     }
42     private cacheString: Array<string> = new Array<string>();
43     private cacheBytes: egret.ByteArray;
44     public fromString(value: string, radix: number = 10): void {
45         if (!value) {
46             this.reset();
47             return;
48         }
49         value = value.toLowerCase();
50         var div: number = 4294967296;
```

```
1      let low: number = 0;
2      var high: number = 0;
3      for (var i: number = 0; i < value.length; i++) {
4          var num: number = value.charCodeAt(i) - 48;
5          if (num > 9)
6              num -= 39;
7          low = low * radix + num;
8          high = high * radix + (low / div >> 0);
9          low = low % div;
10     }
11     this._lowValue = low;
12     this._highValue = high;
13     this.cacheString = [];
14     this.cacheString[radix] = value;
15     this.cacheBytes = null;
16 }
17 public fromBytes(bytes: egret.ByteArray, postion: number = 0): void {
18     try {
19         bytes.position = postion;
20         if (bytes.endian == egret.Endian.LITTLE_ENDIAN) {
21             this._lowValue = bytes.readUnsignedInt();
22             this._highValue = bytes.readUnsignedInt();
23         }
24         else {
25             this._highValue = bytes.readUnsignedInt();
26             this._lowValue = bytes.readUnsignedInt();
27         }
28     }
29     catch (e) {
30         this.reset();
31         return;
32     }
33     this.cacheBytes = null;
34     this.cacheString = [];
35 }
36 private reset(): void {
37     this._highValue = 0;
38     this._lowValue = 0;
39     this.cacheBytes = null;
40     this.cacheString = [];
41 }
42 public clone(): Int64 {
43     return new Int64(this._lowValue, this._highValue);
44 }
45 public copy(value: Int64): void {
46     this.reset();
47     this._lowValue = value._lowValue;
48     this._highValue = value._highValue;
49 }
50 public cloneTo(value: Int64): Int64 {
```

```
1      if (value == null) {
2          value = new Int64();
3      }
4      value.copy(this);
5      return value;
6  }
7  public equals(value: Int64): boolean {
8      if (value == null) return false;
9      return this._highValue == value._highValue && this._lowValue == value._lowValue;
10 }
11 public get bytes(): egret.ByteArray {
12     if (this.cacheBytes)
13         return this.cacheBytes;
14     this.cacheBytes = new egret.ByteArray();
15     this.cacheBytes.endian = egret.Endian.LITTLE_ENDIAN;
16     this.cacheBytes.writeUnsignedInt(this._lowValue);
17     this.cacheBytes.writeUnsignedInt(this._highValue);
18     return this.cacheBytes;
19 }
20 public toNumber(): number
21 {
22     var value:string = this.toString();
23     return value == "" ? 0 : parseInt(value);
24 }
25 public toString(radix: number = 10): string {
26     if (radix < 2 || radix > 36) {
27         throw new RangeError("基数参数必须介于 2 到 36 之间; 当前值为 " + radix
28 + "。");
29     }
30     if (this.cacheString[radix])
31         return this.cacheString[radix];
32     var result: string = "";
33     var lowUint: number = this._lowValue;
34     var highUint: number = this._highValue;
35     var highRemain: number;
36     var lowRemain: number;
37     var tempNum: number;
38     var MaxLowUint: number = Math.pow(2, 32);
39     while (highUint != 0 || lowUint != 0) {
40         highRemain = (highUint % radix);
41         tempNum = highRemain * MaxLowUint + lowUint;
42         lowRemain = tempNum % radix;
43         result = lowRemain.toString(radix) + result;
44         highUint = (highUint - highRemain) / radix;
45         lowUint = (tempNum - lowRemain) / radix;
46     }
47     this.cacheString[radix] = result == "" ? "0" : result;
48     return this.cacheString[radix];
49 }
50 public parseData(data: egret.ByteArray): void {
```



```
1         this._highValue = data.readUnsignedInt();
2         this._lowValue = data.readUnsignedInt();
3     }
4     public toData(data: egret.ByteArray): void {
5         data.writeUnsignedInt(this._highValue);
6         data.writeUnsignedInt(this._lowValue);
7     }
8     public gc(): void {
9         this.cacheBytes = null;
10        this.cacheString = null;
11    }
12 }
13 class GameConst {
14     public constructor() {
15     }
16 }
17 }
18 enum ERROR {
19     MONEY_NOT_ENOUGH,
20     STORE_NOT_ENOUGH,
21     BUY_ZERO,
22     SELL_ZERO,
23     MARKET_NO_GOOD,
24     MAX_STORE_NUM,
25     NEED_LICIENCE,
26 }
27 enum ACHIVE{
28     RELIVE = 1,
29 }
30 class DataBase {
31     public constructor() {
32     }
33     public static gameState:number;
34     public static gamePackage:number;
35     public static money:number;//Int64;
36     public static debt:number;
37     public static deposit:number;//Int64;
38     public static pow:number;
39     public static times:number;
40     public static maxStoreNum:number;
41     public static fame:number;
42     public static marketGoods:varGoods[];
43     public static storeGoods:varGoods[];
44     public static events:string[];
45     public static achives:number[];
46 }
47 declare class BaseButtonSkin extends eui.Skin{
48 }
49 declare class GameSkin extends eui.Skin{
50 }
```

```
1 declare class MarketItemSkin extends eui.Skin{
2 }
3 declare class RankSkin extends eui.Skin{
4 }
5 declare class StartSkin extends eui.Skin{
6 }
7 declare class StoreItemSkin extends eui.Skin{
8 }
9 declare type ResourceRootSelector<T extends string> = () => T;
10 declare type ResourceTypeSelector = (file: string) => string;
11 declare type ResourceNameSelector = (file: string) => string;
12 declare type ResourceMergerSelector = (file: string) => {
13     path: string;
14     alias: string;
15 };
16 declare module RES {
17     var resourceTypeSelector: ResourceTypeSelector;
18     var resourceNameSelector: ResourceNameSelector;
19     var resourceMergerSelector: ResourceMergerSelector | null;
20     function getResourceInfo(path: string): File | null;
21     function setConfigURL(url: string, root: string): void;
22     interface ResourceInfo {
23         url: string;
24         type: string;
25         root: string;
26         crc32?: string;
27         size?: number;
28         name: string;
29         soundType?: string;
30         scale9grid?: string;
31         groupNames?: string[];
32         extra?: boolean;
33         promise?: Promise<any>;
34     }
35     interface Data {
36         resourceRoot: string;
37         typeSelector: ResourceTypeSelector;
38         mergeSelector: ResourceMergerSelector | null;
39         fileSystem: FileSystem;
40         groups: {
41             [groupName: string]: string[];
42         };
43         alias: {
44             [aliasName: string]: string;
45         };
46     }
47     class ResourceConfig {
48         config: Data;
49         constructor();
50         init(): Promise<void>;
```

```
1      __temp__get__type__via__url(url_or_alias: string): string;
2      getKeyByAlias(aliasName: string): string;
3      createGroup(name: string, keys: Array<string>, override?: boolean): boolean;
4      addSubkey(subkey: string, name: string): void;
5      addAlias(alias: any, key: any): void;
6      getType(key: string): string;
7      addResourceData(data: {
8          name: string;
9          type?: string;
10         url: string;
11         root?: string;
12     }): void;
13     destory(): void;
14 }
15 }
16 declare module RES {
17     class ResourceLoader {
18         private groupTotalDic;
19         private numLoadedDic;
20         private itemListDic;
21         private groupErrorDic;
22         private retryTimesDic;
23         maxRetryTimes: number;
24         private priorityQueue;
25         private reporterDic;
26         private dispatcherDic;
27         private failedList;
28         private loadItemErrorDic;
29         private errorDic;
30         load(list: ResourceInfo[], groupName: string, priority: number, reporter?:
31 PromiseTaskReporter): Promise<any>;
32         private loadingCount;
33         thread: number;
34         private next();
35         private removeGroupName(groupName);
36         private queueIndex;
37         private getOneResourceInfo();
38         loadResource(r: ResourceInfo, p?: RES.processor.Processor): Promise<any>;
39         unloadResource(r: ResourceInfo): Promise<any>;
40     }
41 }
42 declare module RES {
43     var systemPid: number;
44     let checkCancelation: MethodDecorator;
45     function profile(): void;
46     var host: ProcessHost;
47     var config: ResourceConfig;
48     var queue: ResourceLoader;
49     interface ProcessHost {
50         state: {
```

```
1      [index: string]: number;
2    };
3    resourceConfig: ResourceConfig;
4    load: (resource: ResourceInfo, processor?: string | processor.Processor) =>
5    Promise<any>;
6    unload: (resource: ResourceInfo) => Promise<any>;
7    save: (resource: ResourceInfo, data: any) => void;
8    get: (resource: ResourceInfo) => any;
9    remove: (resource: ResourceInfo) => void;
10  }
11  class ResourceManagerError extends Error {
12    static errorMessage: {
13      1001: string;
14      1002: string;
15      1005: string;
16      2001: string;
17      2002: string;
18      2003: string;
19      2004: string;
20      2005: string;
21      2006: string;
22    };
23    private __resource_manager_error__;
24    constructor(code: number, replacer?: Object, replacer2?: Object);
25  }
26 }
27 declare namespace RES {
28   interface PromiseTaskReporter {
29     onProgress?: (current: number, total: number) => void;
30     onCancel?: () => void;
31   }
32 }
33 declare module RES {
34   let checkNull: MethodDecorator;
35   let FEATURE_FLAG: {
36     FIX_DUPLICATE_LOAD: number;
37   };
38   namespace upgrade {
39     function setUpgradeGuideLevel(level: "warning" | "silent"): void;
40   }
41 }
42 declare module RES.processor {
43   interface Processor {
44     onLoadStart(host: ProcessHost, resource: ResourceInfo): Promise<any>;
45     onRemoveStart(host: ProcessHost, resource: ResourceInfo): Promise<any>;
46     getData?(host: ProcessHost, resource: ResourceInfo, key: string, subkey: string): any;
47   }
48   function isSupport(resource: ResourceInfo): Processor;
49   function map(type: string, processor: Processor): void;
50   function getRelativePath(url: string, file: string): string;
```

```
1      var ImageProcessor: Processor;
2      var BinaryProcessor: Processor;
3      var TextProcessor: Processor;
4      var JsonProcessor: Processor;
5      var XMLProcessor: Processor;
6      var CommonJSProcessor: Processor;
7      const SheetProcessor: Processor;
8      var FontProcessor: Processor;
9      var SoundProcessor: Processor;
10     var MovieClipProcessor: Processor;
11     const MergeJSONProcessor: Processor;
12     const ResourceConfigProcessor: Processor;
13     const LegacyResourceConfigProcessor: Processor;
14     var PVRProcessor: Processor;
15     const _map: {
16         [index: string]: Processor;
17     };
18 }
19 declare module RES {
20     interface File {
21         url: string;
22         type: string;
23         name: string;
24         root: string;
25     }
26     interface Dictionary {
27         [file: string]: File | Dictionary;
28     }
29     interface FileSystem {
30         addFile(filename: string, type?: string, root?: string): any;
31         getFile(filename: string): File | null;
32         profile(): void;
33     }
34     class NewFileSystem {
35         private data;
36         constructor(data: Dictionary);
37         profile(): void;
38         addFile(filename: string, type?: string): void;
39         getFile(filename: string): File | null;
40         private reslove(dirpath);
41         private mkdir(dirpath);
42         private exists(dirpath);
43     }
44     var fileSystem: FileSystem;
45 }
46 declare module RES {
47     class ResourceEvent extends egret.Event {
48         static ITEM_LOAD_ERROR: string;
49         static CONFIG_COMPLETE: string;
50         static CONFIG_LOAD_ERROR: string;
```

```
1      static GROUP_PROGRESS: string;
2      static GROUP_COMPLETE: string;
3      static GROUP_LOAD_ERROR: string;
4      constructor(type: string, bubbles?: boolean, cancelable?: boolean);
5      itemsLoaded: number;
6      itemsTotal: number;
7      groupName: string;
8      resItem: ResourceItem;
9  }
10 }
11 declare module RES {
12     namespace ResourceItem {
13         const TYPE_XML: string;
14         const TYPE_IMAGE: string;
15         const TYPE_BIN: string;
16         const TYPE_TEXT: string;
17         const TYPE_JSON: string;
18         const TYPE_SHEET: string;
19         const TYPE_FONT: string;
20         const TYPE_SOUND: string;
21         function convertToResItem(r: ResourceInfo): ResourceItem;
22     }
23     interface ResourceItem extends ResourceInfo {
24         name: string;
25         url: string;
26         type: string;
27         data: ResourceInfo;
28         crc32?: string;
29         size?: number;
30         soundType?: string;
31     }
32 }
33 declare namespace RES {
34     namespace path {
35         const normalize: (filename: string) => string;
36         const basename: (filename: string) => string;
37         const dirname: (path: string) => string;
38     }
39 }
40 declare namespace RES {
41 }
42 declare module RES {
43     type GetResAsyncCallback = (value?: any, key?: string) => any;
44     function registerAnalyzer(type: string, analyzerClass: any): void;
45     function loadConfig(url: string, resourceRoot: string): Promise<void>;
46     function loadGroup(name: string, priority?: number, reporter?: PromiseTaskReporter):
47 Promise<void>;
48     function isGroupLoaded(name: string): boolean;
49     function getGroupByName(name: string): Array<ResourceItem>;
50     function createGroup(name: string, keys: Array<string>, override?: boolean): boolean;
```

```
1      function hasRes(key: string): boolean;
2      function getResAsync(key: string): Promise<any>;
3      function getResAsync(key: string, compFunc: GetResAsyncCallback, thisObject: any): void;
4      function getResByUrl(url: string, compFunc: Function, thisObject: any, type?: string): void;
5      function destroyRes(name: string, force?: boolean): Promise<boolean>;
6      function setMaxLoadingThread(thread: number): void;
7      function setMaxRetryTimes(retry: number): void;
8      function addEventListener(type: string, listener: (event: egret.Event) => void, thisObject:
9 any, useCapture?: boolean, priority?: number): void;
10     function removeEventListener(type: string, listener: (event: egret.Event) => void, thisObject:
11 any, useCapture?: boolean): void;
12     function $addResourceData(data: {
13         name: string;
14         type: string;
15         url: string;
16     }): void;
17     class Resource extends egret.EventDispatcher {
18         loadConfig(): Promise<void>;
19         isGroupLoaded(name: string): boolean;
20         getGroupByName(name: string): Array<ResourceInfo>;
21         loadGroup(name: string, priority?: number, reporter?: PromiseTaskReporter):
22 Promise<any>;
23         private _loadGroup(name, priority?, reporter?);
24         loadResources(keys: string[], reporter?: PromiseTaskReporter): Promise<any>;
25         createGroup(name: string, keys: Array<string>, override?: boolean): boolean;
26         hasRes(key: string): boolean;
27         getRes(resKey: string): any;
28         getResAsync(key: string): Promise<any>;
29         getResAsync(key: string, compFunc: GetResAsyncCallback, thisObject: any): void;
30         getResByUrl(url: string, compFunc: Function, thisObject: any, type?: string):
31 Promise<any> | void;
32         destroyRes(name: string, force?: boolean): Promise<boolean>;
33         setMaxLoadingThread(thread: number): void;
34         setMaxRetryTimes(retry: number): void;
35         addResourceData(data: {
36             name: string;
37             type: string;
38             url: string;
39         }): void;
40     }
41 }
42 class StringUtil {
43     public constructor() {
44     }
45     public static getSwfLangTextFlowVar(StrID: string, valArr: string[]): egret.ITextElement[] {
46         return new egret.HtmlTextParser().parser(StringUtil.getSwfLangStrVar(StrID, valArr));
47     }
48     public static getSwfLangStrVarByID(StrID: string, valArr: string[]): string {
49         if (GameLogic.getInstance().strings == null) {
50             return StrID;
```

```
1      }
2      var data: any = GameLogic.getInstance().strings[StrID];
3
4      if (data == null) {
5          return StrID;
6      }
7      return StringUtil.getSwfLangStrVar(data, valArr);
8  }
9  public static getSwfLangStrVar(strData: string, valArr: string[]): string {
10      var indexpre: number;
11      var indexback: number;
12      var strget: string;
13      indexpre = strData.indexOf("{");
14      indexback = strData.indexOf("}");
15      var nextOffset: number = 0;
16      var firstIndex: number;
17      var strFlagPre: number;
18      var strFlagBack: number;
19      var strFlag: string;
20      while (indexpre != -1 && indexback != -1) {
21          strget = strData.substring(indexpre, indexback + 1);
22
23          firstIndex = strData.indexOf("@", nextOffset);
24          //var number: int = int(strData.charAt(strData.indexOf("@", nextOffset) + 1));
25          var numeric: number = parseInt(strData.substring(firstIndex + 1,
26 strData.indexOf(":", firstIndex))) - 1;
27          if (numeric == NaN) {
28              return "stringError:" + strData;
29          }
30          strFlagPre = strData.indexOf("!#[", nextOffset) + 3;
31          if (strFlagPre > 2) {
32              strFlagBack = strData.indexOf("]@", nextOffset);
33              strFlag = strData.substring(strFlagPre, strFlagBack);
34              valArr[numeric] = StringUtil.getSwfLangStr(strFlag + valArr[numeric]);
35          }
36          var strreplace: string = valArr[numeric].toString();
37          strData = strData.replace(strget, strreplace);
38          nextOffset = indexpre + strreplace.length;
39          indexpre = strData.indexOf("{", nextOffset);
40          indexback = strData.indexOf("}", nextOffset);
41      }
42      return strData;
43  }
44  public static getSwfLangStr(StrID: string): string {
45      if (GameLogic.getInstance().strings == null) {
46          return StrID;
47      }
48      var data: any = GameLogic.getInstance().strings[StrID];
49      if (data == null) {
50          return StrID;
```



```
1      }
2      return data.toString();
3  }
4  }
5  declare namespace egret {
6      class Ease {
7          constructor();
8          static get(amount: number): (t: number) => number;
9          static getPowIn(pow: number): (t: number) => number;
10         static getPowOut(pow: number): (t: number) => number;
11         static getPowInOut(pow: number): (t: number) => number;
12         static quadIn: (t: number) => number;
13         static quadOut: (t: number) => number;
14         static quadInOut: (t: number) => number;
15         static cubicIn: (t: number) => number;
16         static cubicOut: (t: number) => number;
17         static cubicInOut: (t: number) => number;
18         static quartIn: (t: number) => number;
19         static quartOut: (t: number) => number;
20         static quartInOut: (t: number) => number;
21         static quintIn: (t: number) => number;
22         static quintOut: (t: number) => number;
23         static quintInOut: (t: number) => number;
24         static sineIn(t: number): number;
25         static sineOut(t: number): number;
26         static sineInOut(t: number): number;
27         static getBackIn(amount: number): (t: number) => number;
28         static backIn: (t: number) => number;
29         static getBackOut(amount: number): (t: any) => number;
30         static backOut: (t: any) => number;
31         static getBackInOut(amount: number): (t: number) => number;
32         static backInOut: (t: number) => number;
33         static circIn(t: number): number;
34         static circOut(t: number): number;
35         static circInOut(t: number): number;
36         static bounceIn(t: number): number;
37         static bounceOut(t: number): number;
38         static bounceInOut(t: number): number;
39         static getElasticIn(amplitude: number, period: number): (t: number) => number;
40         static elasticIn: (t: number) => number;
41         static getElasticOut(amplitude: number, period: number): (t: number) => number;
42         static elasticOut: (t: number) => number;
43         static getElasticInOut(amplitude: number, period: number): (t: number) => number;
44         static elasticInOut: (t: number) => number;
45     }
46 }
47 declare namespace egret {
48     class Tween extends EventDispatcher {
49         private static NONE;
50         private static LOOP;
```

```
1      private static REVERSE;
2      private static _tweens;
3      private static IGNORE;
4      private static _plugins;
5      private static _inited;
6      private _target;
7      private _useTicks;
8      private ignoreGlobalPause;
9      private loop;
10     private pluginData;
11     private _curQueueProps;
12     private _initQueueProps;
13     private _steps;
14     private paused;
15     private duration;
16     private _prevPos;
17     private position;
18     private _prevPosition;
19     private _stepPosition;
20     private passive;
21     static get(target: any, props?: {
22         loop?: boolean;
23         onChange?: Function;
24         onChangeObj?: any;
25     }, pluginData?: any, override?: boolean): Tween;
26     static removeTweens(target: any): void;
27     static pauseTweens(target: any): void;
28     static resumeTweens(target: any): void;
29     private static tick(timestamp, paused?);
30     private static _lastTime;
31     private static _register(tween, value);
32     static removeAllTweens(): void;
33     constructor(target: any, props: any, pluginData: any);
34     private initialize(target, props, pluginData);
35     setPosition(value: number, actionsMode?: number): boolean;
36     private _runAction(action, startPos, endPos, includeStart?);
37     private _updateTargetProps(step, ratio);
38     setPaused(value: boolean): Tween;
39     private _cloneProps(props);
40     private _addStep(o);
41     private _appendQueueProps(o);
42     private _addAction(o);
43     private _set(props, o);
44     wait(duration: number, passive?: boolean): Tween;
45     to(props: any, duration?: number, ease?: Function): Tween;
46     call(callback: Function, thisObj?: any, params?: any[]): Tween;
47     set(props: any, target?: any): Tween;
48     play(tween?: Tween): Tween;
49     pause(tween?: Tween): Tween;
50     $tick(delta: number): void;
```

```
1      }
2  }
3  declare namespace egret.tween {
4      type EaseType = 'quadIn' | 'quadOut' | 'quadOut' | 'quadInOut' | 'cubicIn' | 'cubicOut' |
5  'cubicInOut' | 'quartIn' | 'quartOut' | 'quartInOut' | 'quintIn' | 'quintOut' | 'quintInOut' | 'sineIn'
6  | 'sineOut' | 'sineInOut' | 'backIn' | 'backOut' | 'backInOut' | 'circln' | 'circOut' | 'circlnOut' |
7  'bounceln' | 'bounceOut' | 'bouncelnOut' | 'elasticIn' | 'elasticOut' | 'elasticInOut';
8      abstract class BasePath extends EventDispatcher {
9          name: string;
10     }
11     class To extends BasePath {
12         props: Object;
13         duration: number;
14         ease: EaseType | Function;
15     }
16     class Wait extends BasePath {
17         duration: number;
18         passive: boolean;
19     }
20     class Set extends BasePath {
21         props: Object;
22     }
23     class Tick extends BasePath {
24         delta: number;
25     }
26     class TweenItem extends EventDispatcher {
27         private tween;
28         constructor();
29         private _props;
30         props: any;
31         private _target;
32         target: any;
33         private _paths;
34         paths: BasePath[];
35         play(position?: number): void;
36         pause(): void;
37         private isStop;
38         stop(): void;
39         private createTween(position);
40         private applyPaths();
41         private applyPath(path);
42         private pathComplete(path);
43     }
44     class TweenGroup extends EventDispatcher {
45         private completeCount;
46         constructor();
47         private _items;
48         items: TweenItem[];
49         private registerEvent(add);
50         play(time?: number): void;
```

```
1      pause(): void;
2      stop(): void;
3      private itemComplete(e);
4  }
5  }
6  declare var global: any;
7  declare var __global: any;
8  declare let __define: any;
9  declare namespace egret {
10     interface IHashObject {
11         hashCode: number;
12     }
13     let $hashCount: number;
14     class HashObject implements IHashObject {
15         readonly hashCode: number;
16     }
17 }
18 declare namespace egret {
19     class EventDispatcher extends HashObject implements IEventDispatcher {
20         constructor(target?: IEventDispatcher);
21         $EventDispatcher: Object;
22         $getEventMap(useCapture?: boolean): any;
23         addEventListener(type: string, listener: Function, thisObject: any, useCapture?:
24 boolean, priority?: number): void;
25         once(type: string, listener: Function, thisObject: any, useCapture?: boolean, priority?:
26 number): void;
27         $addListener(type: string, listener: Function, thisObject: any, useCapture?: boolean,
28 priority?: number, dispatchOnce?: boolean): void;
29         $insertEventBin(list: any[], type: string, listener: Function, thisObject: any,
30 useCapture?: boolean, priority?: number, dispatchOnce?: boolean): boolean;
31         removeEventListener(type: string, listener: Function, thisObject: any, useCapture?:
32 boolean): void;
33         $removeEventBin(list: any[], listener: Function, thisObject: any): boolean;
34         hasEventListener(type: string): boolean;
35         willTrigger(type: string): boolean;
36         dispatchEvent(event: Event): boolean;
37         $notifyListener(event: Event, capturePhase: boolean): boolean;
38         dispatchEventWith(type: string, bubbles?: boolean, data?: any, cancelable?: boolean):
39 boolean;
40     }
41 }
42 declare namespace egret.sys {
43     interface EventBin {
44         type: string;
45         listener: Function;
46         thisObject: any;
47         priority: number;
48         target: IEventDispatcher;
49         useCapture: boolean;
50         dispatchOnce: boolean;
```

```
1      }
2  }
3  declare namespace egret {
4      class Filter extends HashObject {
5          type: string;
6          $id: number;
7          $uniforms: any;
8          protected paddingTop: number;
9          protected paddingBottom: number;
10         protected paddingLeft: number;
11         protected paddingRight: number;
12         $obj: any;
13         constructor();
14         $toJson(): string;
15         protected updatePadding(): void;
16         onPropertyChange(): void;
17     }
18 }
19 declare namespace egret {
20     const enum RenderMode {
21         NONE = 1,
22         FILTER = 2,
23         CLIP = 3,
24         SCROLLRECT = 4,
25     }
26     class DisplayObject extends EventDispatcher {
27         constructor();
28         $nativeDisplayObject: egret_native.NativeDisplayObject;
29         protected createNativeDisplayObject(): void;
30         $hasAddToStage: boolean;
31         $children: DisplayObject[];
32         private $name;
33         name: string;
34         $parent: DisplayObjectContainer;
35         readonly parent: DisplayObjectContainer;
36         $setParent(parent: DisplayObjectContainer): void;
37         $onAddToStage(stage: Stage, nestLevel: number): void;
38         $onRemoveFromStage(): void;
39         $stage: Stage;
40         $nestLevel: number;
41         $useTranslate: boolean;
42         protected $updateUseTransform(): void;
43         readonly stage: Stage;
44         matrix: Matrix;
45         private $matrix;
46         private $matrixDirty;
47         $getMatrix(): Matrix;
48         $setMatrix(matrix: Matrix, needUpdateProperties?: boolean): void;
49         private $concatenatedMatrix;
50         $getConcatenatedMatrix(): Matrix;
```

```
1      private $invertedConcatenatedMatrix;
2      $getInvertedConcatenatedMatrix(): Matrix;
3      $x: number;
4      x: number;
5      $getX(): number;
6      $setX(value: number): boolean;
7      $y: number;
8      y: number;
9      $getY(): number;
10     $setY(value: number): boolean;
11     private $scaleX;
12     scaleX: number;
13     $getScaleX(): number;
14     $setScaleX(value: number): void;
15     private $scaleY;
16     scaleY: number;
17     $getScaleY(): number;
18     $setScaleY(value: number): void;
19     private $rotation;
20     rotation: number;
21     $getRotation(): number;
22     $setRotation(value: number): void;
23     private $skewX;
24     private $skewXdeg;
25     skewX: number;
26     $setSkewX(value: number): void;
27     private $skewY;
28     private $skewYdeg;
29     skewY: number;
30     $setSkewY(value: number): void;
31     width: number;
32     $getWidth(): number;
33     $explicitWidth: number;
34     $setWidth(value: number): void;
35     height: number;
36     $explicitHeight: number;
37     $getHeight(): number;
38     $setHeight(value: number): void;
39     readonly measuredWidth: number;
40     readonly measuredHeight: number;
41     $anchorOffsetX: number;
42     anchorOffsetX: number;
43     $setAnchorOffsetX(value: number): void;
44     $anchorOffsetY: number;
45     anchorOffsetY: number;
46     $setAnchorOffsetY(value: number): void;
47     $visible: boolean;
48     visible: boolean;
49     $setVisible(value: boolean): void;
50     $displayList: egret.sys.DisplayList;
```

```
1      private $cacheAsBitmap;
2      cacheAsBitmap: boolean;
3      $setHasDisplayList(value: boolean): void;
4      $cacheDirty: boolean;
5      $cacheDirtyUp(): void;
6      $alpha: number;
7      alpha: number;
8      $setAlpha(value: number): void;
9      static defaultTouchEnabled: boolean;
10     $touchEnabled: boolean;
11     touchEnabled: boolean;
12     $getTouchEnabled(): boolean;
13     $setTouchEnabled(value: boolean): void;
14     $scrollRect: Rectangle;
15     scrollRect: Rectangle;
16     mask: DisplayObject | Rectangle;
17     private $setMaskRect(value);
18     $filters: Array<Filter | CustomFilter>;
19     filters: Array<Filter | CustomFilter>;
20     getTransformedBounds(targetCoordinateSpace: DisplayObject, resultRect?: Rectangle):
21 Rectangle;
22     getBounds(resultRect?: Rectangle, calculateAnchor?: boolean): egret.Rectangle;
23     $getTransformedBounds(targetCoordinateSpace: DisplayObject, resultRect?:
24 Rectangle): Rectangle;
25     globalToLocal(stageX?: number, stageY?: number, resultPoint?: Point): Point;
26     localToGlobal(localX?: number, localY?: number, resultPoint?: Point): Point;
27     $getOriginalBounds(): Rectangle;
28     $measureChildBounds(bounds: Rectangle): void;
29     $getContentBounds(): Rectangle;
30     $measureContentBounds(bounds: Rectangle): void;
31     $parentDisplayList: egret.sys.DisplayList;
32     $renderNode: sys.RenderNode;
33     $renderDirty: boolean;
34     $getRenderNode(): sys.RenderNode;
35     private updateRenderMode();
36     $renderMode: RenderMode;
37     private $measureFiltersOffset(fromParent);
38     $getConcatenatedMatrixAt(root: DisplayObject, matrix: Matrix): void;
39     $updateRenderNode(): void;
40     hitTestPoint(x: number, y: number, shapeFlag?: boolean): boolean;
41     $addListener(type: string, listener: Function, thisObject: any, useCapture?: boolean,
42 priority?: number, dispatchOnce?: boolean): void;
43     removeEventListener(type: string, listener: Function, thisObject: any, useCapture?:
44 boolean): void;
45     $getPropagationList(target: DisplayObject): DisplayObject[];
46     $dispatchPropagationEvent(event: Event, list: DisplayObject[], targetIndex: number):
47 void;
48     willTrigger(type: string): boolean;
49     }
50 }
```

```
1 declare namespace egret {
2     let $TextureScaleFactor: number;
3     class Texture extends HashObject {
4         constructor();
5         private $textureWidth;
6         readonly textureWidth: number;
7         $getTextureWidth(): number;
8         readonly textureHeight: number;
9         $getTextureHeight(): number;
10        $getScaleBitmapWidth(): number;
11        $getScaleBitmapHeight(): number;
12        $initData(bitmapX: number, bitmapY: number, bitmapWidth: number, bitmapHeight:
13        number, offsetX: number, offsetY: number, textureWidth: number, textureHeight: number,
14        sourceWidth: number, sourceHeight: number, rotated?: boolean): void;
15        getPixel32(x: number, y: number): number[];
16        getPixels(x: number, y: number, width?: number, height?: number): number[];
17        toDataURL(type: string, rect?: egret.Rectangle, encoderOptions?: any): string;
18        dispose(): void;
19    }
20 }
21 declare namespace egret {
22     class Event extends HashObject {
23         static ADDED_TO_STAGE: string;
24         static REMOVED_FROM_STAGE: string;
25         static ENTER_FRAME: string;
26         static LOOP_COMPLETE: string;
27         readonly target: any;
28         $setTarget(target: any): boolean;
29         $isPropagationImmediateStopped: boolean;
30         static dispatchEvent(target: IEventDispatcher, type: string, bubbles?: boolean, data?:
31         any): boolean;
32         static _getPropertyData(EventClass: any): any;
33         static create<T extends Event>(EventClass: {
34             new (type: string, bubbles?: boolean, cancelable?: boolean): T;
35             eventPool?: Event[];
36         }, type: string, bubbles?: boolean, cancelable?: boolean): T;
37         static release(event: Event): void;
38     }
39 }
40 declare let RELEASE: boolean;
41 declare namespace egret {
42     function $error(code: number, ...params: any[]): void;
43     function $warn(code: number, ...params: any[]): void;
44     function getString(code: number, ...params: any[]): string;
45     function $markCannotUse(instance: any, property: string, defaultVale: any): void;
46 }
47 declare namespace egret {
48     class Point extends HashObject {
49         static release(point: Point): void;
50         constructor(x?: number, y?: number);
```



```
1      let $TempPoint: Point;
2  }
3  declare namespace egret {
4      class DisplayObjectContainer extends DisplayObject {
5          static $EVENT_ADD_TO_STAGE_LIST: DisplayObject[];
6          static $EVENT_REMOVE_FROM_STAGE_LIST: DisplayObject[];
7          constructor();
8          addChild(child: DisplayObject): DisplayObject;
9          addChildAt(child: DisplayObject, index: number): DisplayObject;
10         $doAddChild(child: DisplayObject, index: number, notifyListeners?: boolean):
11         DisplayObject;
12         contains(child: DisplayObject): boolean;
13         getChildAt(index: number): DisplayObject;
14         getChildIndex(child: egret.DisplayObject): number;
15         getChildByName(name: string): DisplayObject;
16         $doRemoveChild(index: number, notifyListeners?: boolean): DisplayObject;
17         setChildIndex(child: DisplayObject, index: number): void;
18         private doSetChildIndex(child, index);
19         $measureChildBounds(bounds: Rectangle): void;
20         $touchChildren: boolean;
21         $hitTest(stageX: number, stageY: number): DisplayObject;
22     }
23 }
24     EventDispatcher.prototype.removeEventListener = function (type, listener, thisObject,
25 useCapture) {
26         var values = this.$EventDispatcher;
27         var eventMap = useCapture ? values[2 /* captureEventsMap */] : values[1 /*
28 eventsMap */];
29         var list = eventMap[type];
30         if (!list) {
31             return;
32         }
33         if (values[3 /* notifyLevel */] !== 0) {
34             eventMap[type] = list = list.concat();
35         }
36         this.$removeEventBin(list, listener, thisObject);
37         if (list.length == 0) {
38             eventMap[type] = null;
39         }
40     };
41     EventDispatcher.prototype.$removeEventBin = function (list, listener, thisObject) {
42         var length = list.length;
43         for (var i = 0; i < length; i++) {
44             var bin = list[i];
45             if (bin.listener == listener && bin.thisObject == thisObject && bin.target ==
46 this) {
47                 list.splice(i, 1);
48                 return true;
49             }
50         }
```

```
1         return false;
2     };
3
4     EventDispatcher.prototype.$notifyListener = function (event, capturePhase) {
5         var values = this.$EventDispatcher;
6         var eventMap = capturePhase ? values[2 /* captureEventsMap */] : values[1 /*
7 eventsMap */];
8         var list = eventMap[event.$type];
9         if (!list) {
10             return true;
11         }
12         var length = list.length;
13         if (length == 0) {
14             return true;
15         }
16
17         var onceList = ONCE_EVENT_LIST;
18         values[3 /* notifyLevel */]++;
19         for (var i = 0; i < length; i++) {
20             var eventBin = list[i];
21             eventBin.listener.call(eventBin.thisObject, event);
22             if (eventBin.dispatchOnce) {
23                 onceList.push(eventBin);
24             }
25             if (event.$isPropagationImmediateStopped) {
26                 break;
27             }
28         }
29         values[3 /* notifyLevel */]--;
30         while (onceList.length) {
31             var eventBin = onceList.pop();
32             eventBin.target.removeEventListener(eventBin.type, eventBin.listener,
33 eventBin.thisObject, eventBin.useCapture);
34         }
35         return !event.$isDefaultPrevented;
36     };
37     EventDispatcher.prototype.dispatchEventWith = function (type, bubbles, data,
38 cancelable) {
39         if (bubbles || this.hasEventListener(type)) {
40             var event_1 = egret.Event.create(egret.Event, type, bubbles, cancelable);
41             event_1.data = data;
42             var result = this.dispatchEvent(event_1);
43             egret.Event.release(event_1);
44             return result;
45         }
46         return true;
47     };
48     return EventDispatcher;
49 }(egret.HashObject));
50 egret.EventDispatcher = EventDispatcher;
```

```
1      __reflect(EventDispatcher.prototype, "egret.EventDispatcher", ["egret.IEventDispatcher"]);
2  })(egret || (egret = {}));
3  var egret;
4  (function (egret) {
5      var Filter = (function (_super) {
6          __extends(Filter, _super);
7          function Filter() {
8              var _this = _super.call(this) || this;
9              _this.type = null;
10             _this.$id = null;
11             _this.paddingTop = 0;
12             _this.paddingBottom = 0;
13             _this.paddingLeft = 0;
14             _this.paddingRight = 0;
15             _this.$uniforms = {};
16             if (egret.nativeRender) {
17                 egret_native.NativeDisplayObject.createFilter(_this);
18             }
19             return _this;
20         }
21         Filter.prototype.$toJson = function () {
22             return "";
23         };
24         Filter.prototype.updatePadding = function () {
25         };
26         Filter.prototype.onPropertyChange = function () {
27             var self = this;
28             self.updatePadding();
29             if (egret.nativeRender) {
30                 egret_native.NativeDisplayObject.setFilterPadding(self.$id, self.paddingTop,
31 self.paddingBottom, self.paddingLeft, self.paddingRight);
32                 egret_native.NativeDisplayObject.setDataToFilter(self);
33             }
34         };
35         return Filter;
36     })(egret.HashObject));
37     egret.Filter = Filter;
38     __reflect(Filter.prototype, "egret.Filter");
39 })(egret || (egret = {}));
40 var egret;
41 (function (egret) {
42     function clampRotation(value) {
43         value %= 360;
44         if (value > 180) {
45             value -= 360;
46         }
47         else if (value < -180) {
48             value += 360;
49         }
50         return value;
```

```
1      }
2      var DisplayObject = (function (_super) {
3          __extends(DisplayObject, _super);
4          function DisplayObject() {
5              var _this = _super.call(this) || this;
6              _this.$children = null;
7              _this.$name = "";
8              _this.$parent = null;
9              _this.$stage = null;
10             _this.$nestLevel = 0;
11             _this.$useTranslate = false;
12             _this.$matrix = new egret.Matrix();
13             _this.$matrixDirty = false;
14             _this.$x = 0;
15             _this.$y = 0;
16             _this.$scaleX = 1;
17             _this.$scaleY = 1;
18             _this.$rotation = 0;
19             _this.$skewX = 0;
20             _this.$skewXdeg = 0;
21             _this.$skewY = 0;
22             _this.$skewYdeg = 0;
23             _this.$explicitWidth = NaN;
24             _this.$explicitHeight = NaN;
25             _this.$anchorOffsetX = 0;
26             _this.$anchorOffsetY = 0;
27             _this.$visible = true;
28             _this.$displayList = null;
29             _this.$cacheAsBitmap = false;
30             _this.$cacheDirty = false;
31             _this.$alpha = 1;
32             _this.$touchEnabled = DisplayObject.defaultTouchEnabled;
33             _this.$scrollRect = null;
34             _this.$blendMode = 0;
35             _this.$maskedObject = null;
36             _this.$mask = null;
37             _this.$parentDisplayList = null;
38             _this.$renderNode = null;
39             _this.$renderDirty = false;
40             _this.$renderMode = null;
41             if (egret.nativeRender) {
42                 _this.createNativeDisplayObject();
43             }
44             return _this;
45         }
46         DisplayObject.prototype.createNativeDisplayObject = function () {
47             this.$nativeDisplayObject = new egret_native.NativeDisplayObject(0 /*
48 CONTAINER */);
49         };
50         Object.defineProperty(DisplayObject.prototype, "name", {
```

```
1      get: function () {
2          return this.$name;
3      },
4      set: function (value) {
5          this.$name = value;
6      },
7      enumerable: true,
8      configurable: true
9  });
10  Object.defineProperty(DisplayObject.prototype, "parent", {
11      get: function () {
12          return this.$parent;
13      },
14      enumerable: true,
15      configurable: true
16  });
17  DisplayObject.prototype.$setParent = function (parent) {
18      this.$parent = parent;
19  };
20  DisplayObject.prototype.$onAddToStage = function (stage, nestLevel) {
21      var self = this;
22      self.$stage = stage;
23      self.$nestLevel = nestLevel;
24      self.$hasAddToStage = true;
25      egret.Sprite.$EVENT_ADD_TO_STAGE_LIST.push(self);
26  };
27  DisplayObject.prototype.$onRemoveFromStage = function () {
28      var self = this;
29      self.$nestLevel = 0;
30      egret.Sprite.$EVENT_REMOVE_FROM_STAGE_LIST.push(self);
31  };
32  DisplayObject.prototype.$updateUseTransform = function () {
33      var self = this;
34      if (self.$scaleX == 1 && self.$scaleY == 1 && self.$skewX == 0 && self.$skewY == 0)
35  {
36          self.$useTranslate = false;
37      }
38      else {
39          self.$useTranslate = true;
40      }
41  };
42  Object.defineProperty(DisplayObject.prototype, "stage", {
43      get: function () {
44          return this.$stage;
45      },
46      enumerable: true,
47      configurable: true
48  });
49  Object.defineProperty(DisplayObject.prototype, "matrix", {
50      get: function () {
```

```
1         return this.$getMatrix().clone();
2     },
3     set: function (value) {
4         this.$setMatrix(value);
5     },
6     enumerable: true,
7     configurable: true
8 });
9 DisplayObject.prototype.$getMatrix = function () {
10     var self = this;
11     if (self.$matrixDirty) {
12         self.$matrixDirty = false;
13         self.$matrix.$updateScaleAndRotation(self.$scaleX, self.$scaleY, self.$skewX,
14 self.$skewY);
15     }
16     self.$matrix.tx = self.$x;
17     self.$matrix.ty = self.$y;
18     return self.$matrix;
19 };
20 DisplayObject.prototype.$setMatrix = function (matrix, needUpdateProperties) {
21     if (needUpdateProperties === void 0) { needUpdateProperties = true; }
22     var self = this;
23     var m = self.$matrix;
24     m.a = matrix.a;
25     m.b = matrix.b;
26     m.c = matrix.c;
27     m.d = matrix.d;
28     self.$x = matrix.tx;
29     self.$y = matrix.ty;
30     self.$matrixDirty = false;
31     if (m.a == 1 && m.b == 0 && m.c == 0 && m.d == 1) {
32         self.$useTranslate = false;
33     }
34     else {
35         self.$useTranslate = true;
36     }
37     if (needUpdateProperties) {
38         self.$scaleX = m.$getScaleX();
39         self.$scaleY = m.$getScaleY();
40         self.$skewX = matrix.$getSkewX();
41         self.$skewY = matrix.$getSkewY();
42         self.$skewXdeg = clampRotation(self.$skewX * 180 / Math.PI);
43         self.$skewYdeg = clampRotation(self.$skewY * 180 / Math.PI);
44         self.$rotation = clampRotation(self.$skewY * 180 / Math.PI);
45     }
46     if (egret.nativeRender) {
47         self.$nativeDisplayObject.setMatrix(matrix.a, matrix.b, matrix.c, matrix.d,
48 matrix.tx, matrix.ty);
49     }
50 };
```

```
1      DisplayObject.prototype.$getConcatenatedMatrix = function () {
2          var self = this;
3          var matrix = self.$concatenatedMatrix;
4          if (!matrix) {
5              matrix = self.$concatenatedMatrix = new egret.Matrix();
6          }
7          if (self.$parent) {
8              self.$parent.$getConcatenatedMatrix().$preMultiplyInto(self.$getMatrix(),
9 matrix);
10         }
11         else {
12             matrix.copyFrom(self.$getMatrix());
13         }
14         var offsetX = self.$anchorOffsetX;
15         var offsetY = self.$anchorOffsetY;
16         var rect = self.$scrollRect;
17         if (rect) {
18             matrix.$preMultiplyInto(egret.$TempMatrix.setTo(1, 0, 0, 1, -rect.x - offsetX,
19 -rect.y - offsetY), matrix);
20         }
21         else if (offsetX != 0 || offsetY != 0) {
22             matrix.$preMultiplyInto(egret.$TempMatrix.setTo(1, 0, 0, 1, -offsetX,
23 -offsetY), matrix);
24         }
25         return self.$concatenatedMatrix;
26     };
27     DisplayObject.prototype.$getInvertedConcatenatedMatrix = function () {
28         var self = this;
29         if (!self.$invertedConcatenatedMatrix) {
30             self.$invertedConcatenatedMatrix = new egret.Matrix();
31         }
32         self.$getConcatenatedMatrix().$invertInto(self.$invertedConcatenatedMatrix);
33         return self.$invertedConcatenatedMatrix;
34     };
35     Object.defineProperty(DisplayObject.prototype, "x", {
36         get: function () {
37             return this.$getX();
38         },
39         set: function (value) {
40             this.$setX(value);
41         },
42         enumerable: true,
43         configurable: true
44     });
45     DisplayObject.prototype.$getX = function () {
46         return this.$x;
47     };
48     DisplayObject.prototype.$setX = function (value) {
49         var self = this;
50         if (self.$x == value) {
```

```
1         return false;
2     }
3     self.$x = value;
4     if (egret.nativeRender) {
5         self.$nativeDisplayObject.setX(value);
6     }
7     else {
8         var p = self.$parent;
9         if (p && !p.$cacheDirty) {
10             p.$cacheDirty = true;
11             p.$cacheDirtyUp();
12         }
13         var maskedObject = self.$maskedObject;
14         if (maskedObject && !maskedObject.$cacheDirty) {
15             maskedObject.$cacheDirty = true;
16             maskedObject.$cacheDirtyUp();
17         }
18     }
19     return true;
20 };
21 Object.defineProperty(DisplayObject.prototype, "y", {
22     get: function () {
23         return this.$getY();
24     },
25     set: function (value) {
26         this.$setY(value);
27     },
28     enumerable: true,
29     configurable: true
30 });
31 DisplayObject.prototype.$getY = function () {
32     return this.$y;
33 };
34 DisplayObject.prototype.$setY = function (value) {
35     var self = this;
36     if (self.$y == value) {
37         return false;
38     }
39     self.$y = value;
40     if (egret.nativeRender) {
41         self.$nativeDisplayObject.setY(value);
42     }
43     else {
44         var p = self.$parent;
45         if (p && !p.$cacheDirty) {
46             p.$cacheDirty = true;
47             p.$cacheDirtyUp();
48         }
49         var maskedObject = self.$maskedObject;
50         if (maskedObject && !maskedObject.$cacheDirty) {
```



```
1         maskedObject.$cacheDirty = true;
2         maskedObject.$cacheDirtyUp();
3     }
4 }
5     return true;
6 };
7 Object.defineProperty(DisplayObject.prototype, "scaleX", {
8     get: function () {
9         return this.$getScaleX();
10    },
11    set: function (value) {
12        this.$setScaleX(value);
13    },
14    enumerable: true,
15    configurable: true
16 });
17 DisplayObject.prototype.$getScaleX = function () {
18     return this.$scaleX;
19 };
20 DisplayObject.prototype.$setScaleX = function (value) {
21     var self = this;
22     self.$scaleX = value;
23     self.$matrixDirty = true;
24     self.$updateUseTransform();
25     if (egret.nativeRender) {
26         self.$nativeDisplayObject.setScaleX(value);
27     }
28     else {
29         var p = self.$parent;
30         if (p && !p.$cacheDirty) {
31             p.$cacheDirty = true;
32             p.$cacheDirtyUp();
33         }
34         var maskedObject = self.$maskedObject;
35         if (maskedObject && !maskedObject.$cacheDirty) {
36             maskedObject.$cacheDirty = true;
37             maskedObject.$cacheDirtyUp();
38         }
39     }
40 };
41 Object.defineProperty(DisplayObject.prototype, "scaleY", {
42     get: function () {
43         return this.$getScaleY();
44     },
45     set: function (value) {
46         this.$setScaleY(value);
47     },
48     enumerable: true,
49     configurable: true
50 });
```

```
1      DisplayObject.prototype.$getScaleY = function () {
2          return this.$scaleY;
3      };
4      DisplayObject.prototype.$setScaleY = function (value) {
5          var self = this;
6          self.$scaleY = value;
7          self.$matrixDirty = true;
8          self.$updateUseTransform();
9          if (egret.nativeRender) {
10             self.$nativeDisplayObject.setScaleY(value);
11         }
12         else {
13             var p = self.$parent;
14             if (p && !p.$cacheDirty) {
15                 p.$cacheDirty = true;
16                 p.$cacheDirtyUp();
17             }
18             var maskedObject = self.$maskedObject;
19             if (maskedObject && !maskedObject.$cacheDirty) {
20                 maskedObject.$cacheDirty = true;
21                 maskedObject.$cacheDirtyUp();
22             }
23         }
24     };
25     Object.defineProperty(DisplayObject.prototype, "rotation", {
26         get: function () {
27             return this.$getRotation();
28         },
29         set: function (value) {
30             this.$setRotation(value);
31         },
32         enumerable: true,
33         configurable: true
34     });
35     DisplayObject.prototype.$getRotation = function () {
36         return this.$rotation;
37     };
38     DisplayObject.prototype.$setRotation = function (value) {
39         value = clampRotation(value);
40         var self = this;
41         if (value == self.$rotation) {
42             return;
43         }
44         var delta = value - self.$rotation;
45         var angle = delta / 180 * Math.PI;
46         self.$skewX += angle;
47         self.$skewY += angle;
48         self.$rotation = value;
49         self.$matrixDirty = true;
50         self.$updateUseTransform();
```

```
1      if (egret.nativeRender) {
2          self.$nativeDisplayObject.setRotation(value);
3      }
4      else {
5          var p = self.$parent;
6          if (p && !p.$cacheDirty) {
7              p.$cacheDirty = true;
8              p.$cacheDirtyUp();
9          }
10         var maskedObject = self.$maskedObject;
11         if (maskedObject && !maskedObject.$cacheDirty) {
12             maskedObject.$cacheDirty = true;
13             maskedObject.$cacheDirtyUp();
14         }
15     }
16 };
17 Object.defineProperty(DisplayObject.prototype, "skewX", {
18     get: function () {
19         return this.$skewXdeg;
20     },
21     set: function (value) {
22         this.$setSkewX(value);
23     },
24     enumerable: true,
25     configurable: true
26 });
27 DisplayObject.prototype.$setSkewX = function (value) {
28     var self = this;
29     if (value == self.$skewXdeg) {
30         return;
31     }
32     self.$skewXdeg = value;
33     value = clampRotation(value);
34     value = value / 180 * Math.PI;
35     self.$skewX = value;
36     self.$matrixDirty = true;
37     self.$updateUseTransform();
38     if (egret.nativeRender) {
39         self.$nativeDisplayObject.setSkewX(self.$skewXdeg);
40     }
41     else {
42         var p = self.$parent;
43         if (p && !p.$cacheDirty) {
44             p.$cacheDirty = true;
45             p.$cacheDirtyUp();
46         }
47         var maskedObject = self.$maskedObject;
48         if (maskedObject && !maskedObject.$cacheDirty) {
49             maskedObject.$cacheDirty = true;
50             maskedObject.$cacheDirtyUp();
51         }
52     }
53 }
```

```
1         }
2     }
3 };
4 Object.defineProperty(DisplayObject.prototype, "skewY", {
5     get: function () {
6         return this.$skewYdeg;
7     },
8     set: function (value) {
9         this.$setSkewY(value);
10    },
11    enumerable: true,
12    configurable: true
13 });
14 DisplayObject.prototype.$setSkewY = function (value) {
15     var self = this;
16     if (value == self.$skewYdeg) {
17         return;
18     }
19     self.$skewYdeg = value;
20     value = clampRotation(value);
21     value = value / 180 * Math.PI;
22     self.$skewY = value;
23     self.$matrixDirty = true;
24     self.$updateUseTransform();
25     if (egret.nativeRender) {
26         self.$nativeDisplayObject.setSkewY(self.$skewYdeg);
27     }
28     else {
29         var p = self.$parent;
30         if (p && !p.$cacheDirty) {
31             p.$cacheDirty = true;
32             p.$cacheDirtyUp();
33         }
34         var maskedObject = self.$maskedObject;
35         if (maskedObject && !maskedObject.$cacheDirty) {
36             maskedObject.$cacheDirty = true;
37             maskedObject.$cacheDirtyUp();
38         }
39     }
40 };
41 Object.defineProperty(DisplayObject.prototype, "width", {
42     get: function () {
43         return this.$getWidth();
44     },
45     set: function (value) {
46         this.$setWidth(value);
47     },
48     enumerable: true,
49     configurable: true
50 });
```

```
1      DisplayObject.prototype.$getWidth = function () {
2          var self = this;
3          return  isNaN(self.$explicitWidth)    ?    self.$getOriginalBounds().width    :
4  self.$explicitWidth;
5      };
6      DisplayObject.prototype.$setWidth = function (value) {
7          this.$explicitWidth = isNaN(value) ? NaN : value;
8      };
9      Object.defineProperty(DisplayObject.prototype, "height", {
10         get: function () {
11             return this.$getHeight();
12         },
13         set: function (value) {
14             this.$setHeight(value);
15         },
16         enumerable: true,
17         configurable: true
18     });
19     DisplayObject.prototype.$getHeight = function () {
20         var self = this;
21         return  isNaN(self.$explicitHeight)    ?    self.$getOriginalBounds().height    :
22  self.$explicitHeight;
23     };
24     DisplayObject.prototype.$setHeight = function (value) {
25         this.$explicitHeight = isNaN(value) ? NaN : value;
26     };
27     Object.defineProperty(DisplayObject.prototype, "measuredWidth", {
28         get: function () {
29             return this.$getOriginalBounds().width;
30         },
31         enumerable: true,
32         configurable: true
33     });
34     Object.defineProperty(DisplayObject.prototype, "measuredHeight", {
35         get: function () {
36             return this.$getOriginalBounds().height;
37         },
38         enumerable: true,
39         configurable: true
40     });
41     Object.defineProperty(DisplayObject.prototype, "anchorOffsetX", {
42         get: function () {
43             return this.$anchorOffsetX;
44         },
45         set: function (value) {
46             this.$setAnchorOffsetX(value);
47         },
48         enumerable: true,
49         configurable: true
50     });
```

```
1      DisplayObject.prototype.$setAnchorOffsetX = function (value) {
2          var self = this;
3          self.$anchorOffsetX = value;
4          if (egret.nativeRender) {
5              self.$nativeDisplayObject.setAnchorOffsetX(value);
6          }
7      };
8      Object.defineProperty(DisplayObject.prototype, "anchorOffsetY", {
9          get: function () {
10              return this.$anchorOffsetY;
11          },
12          set: function (value) {
13              this.$setAnchorOffsetY(value);
14          },
15          enumerable: true,
16          configurable: true
17      });
18      DisplayObject.prototype.$setAnchorOffsetY = function (value) {
19          var self = this;
20          self.$anchorOffsetY = value;
21          if (egret.nativeRender) {
22              self.$nativeDisplayObject.setAnchorOffsetY(value);
23          }
24      };
25      Object.defineProperty(DisplayObject.prototype, "visible", {
26          get: function () {
27              return this.$visible;
28          },
29          set: function (value) {
30              this.$setVisible(value);
31          },
32          enumerable: true,
33          configurable: true
34      });
35      DisplayObject.prototype.$setVisible = function (value) {
36          var self = this;
37          self.$visible = value;
38          if (egret.nativeRender) {
39              self.$nativeDisplayObject.setVisible(value);
40          }
41          else {
42              self.updateRenderMode();
43              var p = self.$parent;
44              if (p && !p.$cacheDirty) {
45                  p.$cacheDirty = true;
46                  p.$cacheDirtyUp();
47              }
48              var maskedObject = self.$maskedObject;
49              if (maskedObject && !maskedObject.$cacheDirty) {
50                  maskedObject.$cacheDirty = true;
```

```
1         maskedObject.$cacheDirtyUp();
2     }
3 }
4 };
5 Object.defineProperty(DisplayObject.prototype, "cacheAsBitmap", {
6     get: function () {
7         return this.$cacheAsBitmap;
8     },
9     set: function (value) {
10         var self = this;
11         self.$cacheAsBitmap = value;
12         if (egret.nativeRender) {
13             self.$nativeDisplayObject.setCacheAsBitmap(value);
14         }
15         else {
16             self.$setHasDisplayList(value);
17         }
18     },
19     enumerable: true,
20     configurable: true
21 });
22 DisplayObject.prototype.$setHasDisplayList = function (value) {
23     var self = this;
24     var hasDisplayList = !!self.$displayList;
25     if (hasDisplayList == value) {
26         return;
27     }
28     if (value) {
29         var displayList = egret.sys.DisplayList.create(self);
30         if (displayList) {
31             self.$displayList = displayList;
32             self.$cacheDirty = true;
33         }
34     }
35     else {
36         self.$displayList = null;
37     }
38 };
39 DisplayObject.prototype.$cacheDirtyUp = function () {
40     var p = this.$parent;
41     if (p && !p.$cacheDirty) {
42         p.$cacheDirty = true;
43         p.$cacheDirtyUp();
44     }
45 };
46 Object.defineProperty(DisplayObject.prototype, "alpha", {
47     get: function () {
48         return this.$alpha;
49     },
50     set: function (value) {
```

```
1         this.$setAlpha(value);
2     },
3     enumerable: true,
4     configurable: true
5 });
6 DisplayObject.prototype.$setAlpha = function (value) {
7     var self = this;
8     self.$alpha = value;
9     if (egret.nativeRender) {
10         self.$nativeDisplayObject.setAlpha(value);
11     }
12     else {
13         self.updateRenderMode();
14         var p = self.$parent;
15         if (p && !p.$cacheDirty) {
16             p.$cacheDirty = true;
17             p.$cacheDirtyUp();
18         }
19         var maskedObject = self.$maskedObject;
20         if (maskedObject && !maskedObject.$cacheDirty) {
21             maskedObject.$cacheDirty = true;
22             maskedObject.$cacheDirtyUp();
23         }
24     }
25 };
26 Object.defineProperty(DisplayObject.prototype, "touchEnabled", {
27     get: function () {
28         return this.$getTouchEnabled();
29     },
30     set: function (value) {
31         this.$setTouchEnabled(value);
32     },
33     enumerable: true,
34     configurable: true
35 });
36 DisplayObject.prototype.$getTouchEnabled = function () {
37     return this.$touchEnabled;
38 };
39 DisplayObject.prototype.$setTouchEnabled = function (value) {
40     this.$touchEnabled = value;
41 };
42 Object.defineProperty(DisplayObject.prototype, "scrollRect", {
43     get: function () {
44         return this.$scrollRect;
45     },
46     set: function (value) {
47         this.$setScrollRect(value);
48     },
49     enumerable: true,
50     configurable: true
```



```
1      });
2      DisplayObject.prototype.$setScrollRect = function (value) {
3          var self = this;
4          if (!value && !self.$scrollRect) {
5              self.updateRenderMode();
6              return;
7          }
8          if (value) {
9              if (!self.$scrollRect) {
10                 self.$scrollRect = new egret.Rectangle();
11             }
12             self.$scrollRect.copyFrom(value);
13             if (egret.nativeRender) {
14                 self.$nativeDisplayObject.setScrollRect(value.x, value.y, value.width,
15 value.height);
16             }
17         }
18         else {
19             self.$scrollRect = null;
20             if (egret.nativeRender) {
21                 self.$nativeDisplayObject.setScrollRect(0, 0, 0, 0);
22             }
23         }
24         if (!egret.nativeRender) {
25             self.updateRenderMode();
26         }
27     };
28     Object.defineProperty(DisplayObject.prototype, "blendMode", {
29         get: function () {
30             return egret.sys.numberToBlendMode(this.$blendMode);
31         },
32         set: function (value) {
33             var self = this;
34             var mode = egret.sys.blendModeToNumber(value);
35             self.$blendMode = mode;
36             if (egret.nativeRender) {
37                 self.$nativeDisplayObject.setBlendMode(mode);
38             }
39             else {
40                 self.updateRenderMode();
41                 var p = self.$parent;
42                 if (p && !p.$cacheDirty) {
43                     p.$cacheDirty = true;
44                     p.$cacheDirtyUp();
45                 }
46                 var maskedObject = self.$maskedObject;
47                 if (maskedObject && !maskedObject.$cacheDirty) {
48                     maskedObject.$cacheDirty = true;
49                     maskedObject.$cacheDirtyUp();
50                 }
51             }
52         }
53     });
```



```
1         if (!egret.nativeRender) {
2             self.$mask.updateRenderMode();
3         }
4     }
5     if (self.mask) {
6         if (egret.nativeRender) {
7             self.$nativeDisplayObject.setMask(-1);
8         }
9         self.$mask = null;
10    }
11    }
12    }
13    else {
14        if (self.$mask) {
15            self.$mask.$maskedObject = null;
16            if (!egret.nativeRender) {
17                self.$mask.updateRenderMode();
18            }
19        }
20        if (self.mask) {
21            if (egret.nativeRender) {
22                self.$nativeDisplayObject.setMask(-1);
23            }
24            self.$mask = null;
25        }
26        if (self.$maskRect) {
27            if (egret.nativeRender) {
28                self.$nativeDisplayObject.setMaskRect(0, 0, 0, 0);
29            }
30            self.$maskRect = null;
31        }
32    }
33    if (!egret.nativeRender) {
34        self.updateRenderMode();
35    }
36    },
37    enumerable: true,
38    configurable: true
39    });
40    var egret;
41    (function (egret) {
42        var BlurFilter = (function (_super) {
43            __extends(BlurFilter, _super);
44            function BlurFilter(blurX, blurY, quality) {
45                if (blurX === void 0) { blurX = 4; }
46                if (blurY === void 0) { blurY = 4; }
47                if (quality === void 0) { quality = 1; }
48                var _this = _super.call(this) || this;
49                var self = _this;
50                self.type = "blur";
```

```
1         self.$blurX = blurX;
2         self.$blurY = blurY;
3         self.$quality = quality;
4         self.blurXFilter = new BlurXFilter(blurX);
5         self.blurYFilter = new BlurYFilter(blurY);
6         self.onPropertyChange();
7         return _this;
8     }
9     Object.defineProperty(BlurFilter.prototype, "blurX", {
10         get: function () {
11             return this.$blurX;
12         },
13         set: function (value) {
14             var self = this;
15             if (self.$blurX == value) {
16                 return;
17             }
18             self.$blurX = value;
19             self.blurXFilter.blurX = value;
20             self.onPropertyChange();
21         },
22         enumerable: true,
23         configurable: true
24     });
25     Object.defineProperty(BlurFilter.prototype, "blurY", {
26         get: function () {
27             return this.$blurY;
28         },
29         set: function (value) {
30             var self = this;
31             if (self.$blurY == value) {
32                 return;
33             }
34             self.$blurY = value;
35             self.blurYFilter.blurY = value;
36             self.onPropertyChange();
37         },
38         enumerable: true,
39         configurable: true
40     });
41 }
42 class GameEvent {
43     public constructor() {
44     }
45 }
46 class GameUI extends eui.Component {
47     public constructor() {
48         super();
49         this.skinName = "GameSkin";
50     }
```

```
1     private gp_market: eui.Group;
2     private gp_store: eui.Group;
3     private gp_over: eui.Group;
4     private lbl_day: eui.Label;
5     private lbl_store: eui.Label;
6     private lbl_1: eui.Label;
7     private lbl_2: eui.Label;
8     private lbl_3: eui.Label;
9     private lbl_4: eui.Label;
10    private lbl_5: eui.Label;
11    private lbl_6: eui.Label;
12    private rect_evt: eui.Rect;
13    private cb_0:eui.CheckBox;
14    private crtPop: number;
15    private market_arr: MarketItem[];
16    private store_arr: StoreItem[];
17    private gamestate: number;
18    private max_num: number;
19    private data: msgLifeDataRsp;
20    private leftStore: number;
21    protected childrenCreated() {
22        super.childrenCreated();
23        GameLogic.getInstance().gameui = this;
24        this.market_arr = [];
25        this.store_arr = [];
26        this.eventlist = [];
27        this.initView();
28        this.initEvent();
29        GameCommand.getInstance().startGame();
30    }
31    private eventlist: string[];
32    private eventpopping: boolean;
33    public eventAppear(str: string) {
34        if(GameLogic.getInstance().cbSelected){
35            return;
36        }
37        if (this.eventpopping) {
38            this.eventlist.push(str);
39            return;
40        }
41        this.popEvent(str);
42    }
43    private eventNext() {
44        this.eventpopping = false;
45        if (this.eventlist.length > 0) {
46            let str = this.eventlist.shift();
47            this.popEvent(str);
48        }
49        else {
50            this.pop(0);
```

```
1      }
2  }
3  private popEvent(str: string) {
4      this.eventpopping = true;
5      this.pop(11);
6      this['lbl_event_1'].text = str;
7  }
8  public initData(msg: msgLifeDataRsp) {
9      this.data = msg;
10     this.setLeft();
11     this.lbl_1.text = this.data.dwMoney.toString();
12     this.lbl_2.text = this.data.dwDeposit.toString();
13     this.lbl_3.text = this.data.dwDebt.toString();
14     this.lbl_4.text = this.data.dwPow.toString();
15     this.lbl_5.text = this.data.dwFame.toString();
16     let maxday = GameLogic.getInstance().data['maxday'];
17     this.lbl_day.text = (maxday - this.data.dwTimes) + "/" + maxday + "天";
18 }
19 private setLeft() {
20     let n = this.getStoreNum();
21     this.leftStore = this.data.dwMaxStoreNum - n;
22     this.lbl_store.text = n + "/" + this.data.dwMaxStoreNum;
23 }
24 private getStoreNum(): number {
25     let n: number = 0;
26     for (let i: number = 0; i < this.store_arr.length; i++) {
27         let item = this.store_arr[i];
28         n += item.good.dwNum;
29     }
30     return n;
31 }
32 public initMarket(msg: msgGoodsBuyRsp) {
33     this.clearMarket();
34     for (let i: number = 0; i < msg.goods.length; i++) {
35         let item = new MarketItem(msg.goods[i]);
36         item.y = (item.height + 2) * i;
37         item.addEventListener(egret.TouchEvent.TOUCH_TAP, this.clickMarketItem, this);
38         this.market_arr.push(item);
39         this.gp_market.addChild(item);
40     }
41 }
42 private crtMarketItem: MarketItem;
43 private clickMarketItem(e: egret.TouchEvent) {
44     let item = e.currentTarget as MarketItem;
45     if (this.crtMarketItem != null) {
46         this.crtMarketItem.select = false;
47     }
48     this.crtMarketItem = item;
49     this.crtMarketItem.select = true;
50     this.pop(9);
```

```

1      let max = Math.floor(this.data.dwMoney / item.good.dwPrice);
2      this.max_num = max > this.leftStore ? this.leftStore : max;
3      this['lbl_buy_1'].text = item.good.strName;
4      this['lbl_num6'].text = this.max_num + "";
5  }
6  public initStore(msg: msgGoodsStoreRsp): void {
7      this.clearStore();
8      for (let i: number = 0; i < msg.goods.length; i++) {
9          let item = new StoreItem(msg.goods[i]);
10         item.y = (item.height + 2) * i;
11         item.addEventListener(egret.TouchEvent.TOUCH_TAP, this.clickStoreItem, this);
12         this.store_arr.push(item);
13         this.gp_store.addChild(item);
14     }
15     this.setLeft();
16 }
17 private crtStoreItem: StoreItem;
18 private clickStoreItem(e: egret.TouchEvent) {
19     let item = e.currentTarget as StoreItem;
20     if (this.crtStoreItem != null) {
21         this.crtStoreItem.select = false;
22     }
23     this.crtStoreItem = item;
24     this.crtStoreItem.select = true;
25     this.pop(10);
26     this.max_num = this.crtStoreItem.good.dwNum;
27     this['lbl_sell_1'].text = item.good.strName;
28     this['lbl_num7'].text = this.max_num + "";
29 }
30 public storeUp(arr): void {
31 }
32 public over() {
33     this['gp_over'].visible = true;
34     let str:string = "";
35     if (this.data.dwPow <= 0) {
36         str += StringUtil.getSwfLangStr("s20") + "\n";
37         this['btn_27'].visible = false;
38     }
39     else {
40         str += StringUtil.getSwfLangStr("s11") + "\n";
41         str += StringUtil.getSwfLangStrVarById("s21", [DataBase.money + ""]) + "\n";
42         str += StringUtil.getSwfLangStr("s12") + "\n";
43         for (let i: number = 0; i < 5; i++) {
44             str += StringUtil.getSwfLangStrVarById("s1" + (3 + i), [DataBase.achives[i] +
45 ""]) + "\n";
46         }
47         str += StringUtil.getSwfLangStr("s19") + "\n";
48         str += StringUtil.getSwfLangStr("s50") + "\n";
49         this['btn_27'].visible = true;
50     }

```

```
1         this['lbl_over_1'].text = str;
2     }
3     public errorRsp(i: number) {
4         this.eventAppear(StringUtil.getSwfLangStr("e" + i));
5     }
6     private initView() {
7         this.cb_0.selected = GameLogic.getInstance().cbSelected;
8     }
9     private initEvent() {
10        this.addEventListener(egret.Event.REMOVED_FROM_STAGE, this.clear, this);
11        for (let i: number = 0; i <= 27; i++) {
12            let btn: eui.Button = this['btn_' + i];
13            btn.name = i + "";
14            btn.addEventListener(egret.TouchEvent.TOUCH_TAP, this.clickBtn, this);
15        }
16        for (let i: number = 0; i <= 7; i++) {
17            let lbl: eui.Label = this['lbl_num' + i];
18            lbl.name = 'lbl' + i;
19            lbl.addEventListener(egret.Event.CHANGE, this.txtChange, this);
20            lbl.addEventListener(egret.TouchEvent.TOUCH_TAP, this.txtClick, this);
21        }
22        this.cb_0.addEventListener(egret.Event.CHANGE, this.cbChange, this);
23        this.rect_evt.addEventListener(egret.TouchEvent.TOUCH_TAP, this.clickRect, this);
24    }
25    private cbChange(){
26        GameLogic.getInstance().cbSelected = this.cb_0.selected;
27    }
28    private clickRect(e: egret.TouchEvent) {
29        this.eventNext();
30    }
31    private txtClick(e: egret.TouchEvent) {
32        let lbl: eui.Label = e.currentTarget as eui.Label;
33        let i = parseInt(lbl.name.slice(3));
34        switch (i) {
35            case 1://存款
36                this.max_num = this.data.dwMoney;
37                break;
38            case 2://取款
39                this.max_num = this.data.dwDeposit;
40                break;
41        }
42    }
43    private txtChange(e: egret.Event) {
44        let lbl: eui.Label = e.currentTarget as eui.Label;
45        let n = parseInt(lbl.text);
46        if (n > this.max_num) {
47            lbl.text = this.max_num + "";
48        }
49    }
50    private clickBtn(e: egret.TouchEvent) {
```



```
1      let i = parseInt(e.currentTarget.name);
2      if (this.gamestate == 0) {
3          if (i < 9 && i > 12 && i < 26) {
4              return;
5          }
6      }
7      switch (i) {
8          case 0://捐款
9              this.pop(0);
10             GameCommand.getInstance().charity(parseInt(this['lbl_num0'].text));
11             break;
12         case 1:
13         case 2:
14         case 3:
15             this.pop(0);
16             GameCommand.getInstance().passOneDay();
17             break;
18         case 4://慈善
19             this.pop(i);
20             this.max_num = this.data.dwMoney;
21             this['lbl_charity'].text = StringUtil.getSwfLangStr("s10");
22             this['lbl_num0'].text = "0";
23             break;
24         case 5://银行
25             this.pop(i);
26             this['lbl_num1'].text = this.data.dwMoney + "";
27             this['lbl_num2'].text = this.data.dwDeposit + "";
28             break;
29         case 6://医院
30             this.pop(i);
31             let n = 100 - this.data.dwPow;
32             if (n <= 0) {
33                 this['lbl_hos_1'].text = StringUtil.getSwfLangStr("s6");
34                 this.max_num = 0;
35             }
36             else {
37                 this['lbl_hos_1'].text = StringUtil.getSwfLangStrVarByID("s7",
38 [GameLogic.getInstance().data['hospital'] + ""]);
39                 this.max_num = n;
40             }
41             this['lbl_num3'].text = this.max_num + "";
42             break;
43         case 7://中介
44             this.pop(i);
45             let n7 = GameLogic.getInstance().data['maxstore'] -
46 this.data.dwMaxStoreNum;
47             if (n7 <= 0) {
48                 this['lbl_medi_1'].text = StringUtil.getSwfLangStr("s8");
49             }
50             else {
```

```
1         let n70 = GameLogic.getInstance().data['storeprice'];
2         let n71 = Math.floor(Math.random() * n70 / 5) + n70;
3         this['lbl_medi_1'].text = StringUtil.getSwfLangStrVarByID("s9", [n71 +
4     ""]);
5     }
6     break;
7     case 8://邮局
8         this.pop(i);
9         let n8 = this.data.dwDebt;
10        let n80 = this.data.dwMoney > n8 ? n8 : this.data.dwMoney;
11        if (n8 <= 0) {//没有债务
12            this['lbl_post_1'].text = StringUtil.getSwfLangStr("s10");
13            this.max_num = 0;
14        }
15        else {
16            this['lbl_post_1'].text = StringUtil.getSwfLangStr("s11");
17            this.max_num = n80;
18        }
19        this['lbl_num5'].text = this.max_num + "";
20        break;
21    case 9://转发
22        GameLogic.getInstance().share(1);
23        break;
24    case 10://广告
25        break;
26    case 11://排行榜
27        break;
28    case 12://重新开始
29    case 26:
30        this.restart();
31        break;
32    case 13://存钱
33        GameCommand.getInstance().cun(parseInt(this['lbl_num1'].text));
34        this.pop(0);
35        break;
36    case 14://取钱
37        GameCommand.getInstance().qu(parseInt(this['lbl_num2'].text));
38        this.pop(0);
39        break;
40    case 15://治疗
41        if (this.data.dwPow < 100) {
42            GameCommand.getInstance().treat(100 - this.data.dwPow);
43        }
44        this.pop(0);
45        break;
46    case 17://买柜子
47        let n17 = GameLogic.getInstance().data['storeprice'];
48        n17 = Math.floor(Math.random() * n17 / 5) + n17;
49        GameCommand.getInstance().buyStore(n17);
50        this.pop(0);
```

```
1         break;
2     case 19://还债
3         let n19 = parseInt(this['lbl_num5'].text);
4         n19 = this.data.dwMoney < n19 ? this.data.dwMoney : n19;
5         GameCommand.getInstance().huan(n19);
6         this.pop(0);
7         break;
8     case 21://购买
9         let n21 = parseInt(this['lbl_num6'].text);
10        if (n21 > 0) {
11
12            GameCommand.getInstance().buyGoods(this.crtMarketItem.good.dwID, n21);
13        }
14        this.pop(0);
15        break;
16    case 23://出售
17        let n23 = parseInt(this['lbl_num7'].text);
18        if (n23 > 0) {
19            GameCommand.getInstance().sellGoods(this.crtStoreItem.good.dwID,
20 n23);
21        }
22        this.pop(0);
23        break;
24    case 16://关闭
25    case 18:
26    case 20:
27    case 22:
28    case 24:
29        this.pop(0);
30        break;
31    case 25:
32        this.eventNext();
33        break;
34    case 27://炫耀
35        break;
36    }
37    }
38    private pop(i: number) {
39        if (this.crtPop != null) {
40            let gp = this['gp_' + this.crtPop];
41            if (gp != null) {
42                gp.visible = false;
43            }
44        }
45        if (i == 0) {
46            this.eventpopping = false;
47        }
48        let gp = this['gp_' + i];
49        if (gp != null) {
50            gp.visible = true;
```

```
1         this.crtPop = i;
2     }
3     else {
4         this.crtPop = null;
5     }
6 }
7 private clearMarket() {
8     for (let i: number = 0; i < this.market_arr.length; i++) {
9         let item = this.market_arr[i];
10    }
11    this.gp_market.removeChildren();
12    this.market_arr = [];
13    this.crtMarketItem = null;
14 }
15 private clearStore() {
16     for (let i: number = 0; i < this.store_arr.length; i++) {
17         let item = this.store_arr[i];
18         item.removeEventListener(egret.TouchEvent.TOUCH_TAP, this.clickStoreItem,
19 this);
20     }
21
22     this.gp_store.removeChildren();
23     this.store_arr = [];
24     this.crtStoreItem = null;
25 }
26
27 private restart() {
28     GameLogic.getInstance().openStart();
29 }
30 private clear() {
31     this.clearEvent();
32     this.clearMarket();
33     this.clearStore();
34     GameLogic.getInstance().gameui = null;
35 }
36 private clearEvent() {
37     this.removeEventListener(egret.Event.REMOVED_FROM_STAGE, this.clear, this);
38     for (let i: number = 0; i <= 27; i++) {
39         let btn: eui.Button = this['btn_' + i];
40         btn.removeEventListener(egret.TouchEvent.TOUCH_TAP, this.clickBtn, this);
41     }
42     for (let i: number = 0; i <= 7; i++) {
43         let lbl: eui.Label = this['lbl_num' + i];
44         lbl.removeEventListener(egret.Event.CHANGE, this.txtChange, this);
45         lbl.removeEventListener(egret.TouchEvent.TOUCH_TAP, this.txtClick, this);
46     }
47     this.cb_0.removeEventListener(egret.Event.CHANGE, this.cbChange, this);
48     this.rect_evt.removeEventListener(egret.TouchEvent.TOUCH_TAP, this.clickRect, this);
49 }
50 }
```