Kursaufgabe

Ihre Aufgabe im C#-Blockkurs ist es, eine Mission im Flattiverse zu erfüllen und gegen Ende an einem Team-Shoot-The-Flag-Spiel mitzumachen. Bei Flattiverse handelt es sich um einen Multiplayer-Server, zu dem Sie mit einer .NET Assembly (die Connector.dll) verbinden können. Wie Sie den Flattiverse-Connector verwenden, wird nur leider nirgends ausreichend aktuell erklärt:

Wir haben die letzten Jahre den Flattiverse-Server und das eigentliche Spiel stark verbessert und sind nun leider mit der Homepage nicht fertig geworden. Wenn Sie auf der Homepage einen Fehler entdecken oder etwas nicht funktioniert, sprechen Sie bitte einen Dozenten an.

Die vorgeschlagene Reihenfolge für den besten Erfolg im Kurs:

1. Besuchen Sie die Flattiverse-Alpha-Homepage: http://www.flattiverse.com/
2. Registrieren Sie sich und bestätigen Sie die Opt-In-Mail.
3. Lesen Sie sich die ausgeteilte „Schnelleinstieg C#/.NET"-Dokumentation und lesen Sie die unten, bzw. auf der nächsten Seite aufgeführten Ergänzungen.
4. Versuchen Sie zuerst `FlattiverseMessages` anzuzeigen, die Ihnen der Server schickt. Verwenden Sie dafür nach Möglichkeit eine Schrift mit fixer Breite – für den Anfang genügt ein Konsolenprojekt.
5. In der MOTD befinden sich weitere Anweisungen. Die MOTD ist eine Message, die Ihnen vom Server übermittelt wird.
6. Beginnen Sie spätestens jetzt ein Forms- oder GTK-Projekt.
7. Am Ende des 3. Kurstages schicken Sie Screenshots Ihrer Anwendung im PNG-Format zusammen mit Ihrem Account-Namen in Flattiverse und ihrem Kürzel an der HS-Esslingen an info@flattiverse.com. Auch wenn Sie kein Studierender der Hochschule Esslingen sind schicken Sie bitte Screenshots.

Unterschiede zwischen Dokumentation und aktuellem Softwarestand

Hier finden Sie alle Unterschiede zwischen der Dokumentation und dem aktuellen Softwarestand, sofern ich daran gedacht habe diesen zu erwähnen:

1. Ein Objekt der Klasse `UniverseGroup` besitzt eine `Join`-Funktion mit entweder zwei oder vier Parametern. Die Dokumentation dokumentiert an dieser Stelle nur einen Parameter. Die Parameter bedeuten in Ihrer auftretenden Reihenfolge:
   - `name`: Gibt den Name ihrer Session an. Dieser Name ist später auf der (neuen) Homepage einsehbar.
   - `team`: Hier geben Sie das Team an, dem Sie beitreten möchten. Dies muss ein Team aus `Teams` aus der entsprechenden `UniverseGroup` sein.
   - `clan`: Hier können Sie ein Clan-Tag angeben. Dies ist von anderen Spielern abfragbar und hat sonst keine Auswirkungen. Geben Sie `null` an, wenn Sie hier nichts angeben möchten.
   - `password`: Hier können Sie ein Passwort angeben, falls eine `UniverseGroup` passwortgeschützt ist. Geben Sie `null` an, wenn Sie kein Passwort übermitteln wollen.

2. Die Dokumentation erwähnt die Funktionen `Wait()`, `Commit()` und `PreWait()`. Diese sind sehr essentiell und haben sich maßgeblich verändert, daher hier eine komplette Erklärung:
   - Flattiverse basiert auf Ticks. In einem Tick wird ein Spielzyklus berechnet: Alle Einheiten bewegen sich, Kollisionen werden geprüft und neue Spieler-Aktionen werden ausgeführt. Den fairsten Spielablauf erhalten Sie durch die folgende, in Flattiverse implementierte Reihenfolge. Hier der Komplettdurchlauf eines Zyklus, bzw. Ticks:
     i. Alle Spieler haben `Commit()` aufgerufen oder die Universen-Gruppe hat die maximale Wartezeit überschritten.
     ii. Der Server sperrt die Universen-Gruppe gegen Änderungen und signalisiert allen Teilnehmern (Spielern) `PreWait`. Alle jetzt gesendeten Spielerbefehle an die Universen-Gruppe werden eingereiht und warten.
     iii. Jeder Spieler wird über den Connector informiert, dass der Server jetzt die Universen-Gruppe berechnet. An dieser Stelle besitzen alle Objekte noch die Zustände aus dem letzten Tick, da die neuen Werte erst gerade im Server berechnet werden und noch nicht an die Spieler übermittelt wurden. Befehle, die Spieler jetzt senden werden vom Server gesammelt und nach der Berechnung der Universen-Gruppe verarbeitet.
     iv. Der Server verarbeitet alle im letzten Tick gesendeten Befehle: Zuerst wird jedes der Raumschiffe bewegt, dann werden die `Shoot()`-Aufrufe verarbeitet und die Schüsse erzeugt.
     v. Der Server übermittelt alle neuen Spieldaten (Hülle, Schilde, ist jemand am Leben?, etc.) und signalisiert allen Spieler `Wait`. Anschließend entsperrt der Server die Universen-Gruppe und verarbeitet alle inzwischen angekommenen Befehle.
     vi. Jeder Spieler wird über den Connector informiert, dass die Berechnung der Universen-Gruppe vorbei ist.
     vii. Nachdem ein Spieler alle Befehle, wie `Shoot()`, `Scan()` und `Move()` ausgeführt hat führt er `Commit()` aus.
     viii. Weiter bei i.
   - Wenn Sie einer Universen-Gruppe beitreten wird ihr Connector automatisch für Sie Committen, bis Sie ein `UniverseGroupFlowControl`-Objekt abgeholt haben. Der Connector beginnt auch wieder automatisch für Sie zu Committen, wenn Sie das letzte `UniverseGroupFlowControl`-Objekt zurückgegeben haben (=`.Dispose()` aufgerufen haben).
   - Ein `UniverseGroupFlowControl`-Objekt erhalten Sie von der Universen-Gruppe, der Sie mit `Join()` beigetreten sind: `universeGroup.GetNewFlowControl()`.
   - Mit der Methode `PreWait()` im `UniverseGroupFlowControl`-Objekt warten Sie auf das nächste `PreWait` nach ihrem letzten `Commit()`. Führen Sie `ugfc.PreWait()` mehrmals aus, sind die nachfolgenden Aufrufe redundant. `PreWait()` liefert ein `TimeSpan` zurück, welches Sie darüber informiert, wie lange Sie für die Berechnung des aktuellen Ticks noch Zeit haben.
   - Mit der Methode `Wait()`: Analog zu `PreWait()`, nur für `Wait`.
   - Mit der Methode `Commit()` teilen Sie dem Server mit, dass Sie fertig mit Ihren Berechnungen sind und der nächste Tick beginnen kann. `Commit()` liefert `true` zurück, solange Sie innerhalb ihres Ticks committen.
   - Wenn Sie ein `UniverseGroupFlowControl`-Objekt nicht mehr benötigen, müssen Sie es mit `Dispose()` zurückgeben. Am besten verpacken Sie das in einem `using`-Block.
   - Ein `UniverseGroupFlowControl`-Objekt kann nur von dem Thread aus benutzt werden, aus dem heraus es instanziiert wurde.
   - Sie können beliebig viele `UniverseGroupFlowControl`-Objekte erzeugen.

Die folgenden Tipps sollen Ihnen helfen den Kurs möglichst effektiv zu bestehen und gegen Ende ein möglichst gutes Raumschiff programmiert zu haben. Sie spielen am letzten Tag im Team gegen andere Teams und sollten daher das Beste aus den nächsten Tagen machen. (Oder Sie geben einfach auf, wie 80% der Kursteilnehmer.)

1. Bei Fragen, fragen Sie zuerst ihre Mitkommilitonen oder Mitkommilitoninnen.
2. Im Kurs befinden sich Wiederholer des Kurses. Leute, denen dieser Kurs Spaß macht und die sich sehr gut auskennen. Fragen Sie diese, falls Ihnen die Erstteilnehmer nicht weiter helfen können.
3. Ziehen Sie sich die Tipps aus der MOTD zu Rate.
4. Sehen Sie in den Übungsvideos nach.
5. Wenn Sie ein Raumschiff registrieren möchten steht Ihnen die vorkonfigurierte Klasse `@Ship` zur Verfügung.
6. Wenn Sie eine Frage zur API haben hilft Ihnen möglicherweise der Objektkatalog weiter. Unter Visual Studio ist dieser in der Regel mit `Strg+Alt+J` erreichbar.
7. Benutzen Sie Google.
8. Wenn Sie ihre Frage oder ihr Problem durch die vorherigen Punkte nicht lösen konnten, fragen Sie bitte Prof. Harald Melcher.

Weitere Tipps:

1. Aufgrund von Schrifteigenheiten kann es sein, dass wenn Sie etwas aus einer PDF herauskopieren und dies in Visual Studio wieder einfügen sich einzelne Zeichen verändern. Bevor Sie sich beschweren: Tippen Sie Universen-Namen einfach ab, wenn etwas nicht klappt.