


**\*\*You:\*\***

你将使用SMAC和PYSC2等相关库编写一个Starcraft2 AI，请做好准备。

**\*\*ChatGPT:\*\***

好的，我已经准备好帮助你编写一个Starcraft2 AI。首先，我们需要确保你已经安装了所有必要的库，包括SMAC和PYSC2。这些库可以通过pip进行安装：

```
bash  Copy code
pip install sc2
pip install pysc2
pip install smac
```

然后，我们可以开始编写AI的基本框架。在Starcraft2中，AI的主要任务包括决策制定、单位管理和战术执行等。我们可以从创建一个基本的Agent开始，然后逐步添加更复杂的功能。

这是一个基本的Starcraft2 AI的框架：

```
python  Copy code
from pysc2.agents import base_agent
from pysc2.lib import actions, features, units
from absl import app

class BasicAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(BasicAgent, self).step(obs)

        # 在这里添加你的代码

def main(unused_argv):
    agent = BasicAgent()
    try:
        while True:
            with sc2_env.SC2Env(
                map_name="Simple64",
                players=[sc2_env.Agent(sc2_env.Race.terran),
                        sc2_env.Bot(sc2_env.Race.random,
                                    sc2_env.Difficulty.very_easy)],
                agent_interface_format=features.AgentInterfaceFormat(
                    feature_dimensions=features.Dimensions(screen=84, minimap=64),
                    use_feature_units=True),
                step_mul=16,
                game_steps_per_episode=0,
                visualize=True) as env:

                agent.setup(env.observation_spec(), env.action_spec())

                timesteps = env.reset()
                agent.reset()

                while True:
                    step_actions = [agent.step(timesteps[0])]
                    if timesteps[0].last():
                        break
                    timesteps = env.step(step_actions)

            except KeyboardInterrupt:
                pass

if __name__ == "__main__":
    app.run(main)
```

这个AI目前还没有任何功能，你需要在step函数中添加你的代码来定义AI的行为。例如，你可以让AI创建更多的单位，攻击敌人，或者收集资源等。

如果你需要更多的帮助，例如如何使用SMAC库，或者如何编写更复杂的AI，请告诉我！


**\*\*You:\*\***

没错，请给予我更多编写更复杂AI的帮助！首先，1.从让AI能够创建所有Terran的单位和建筑，2.单位能施放所有可用技能。开始吧

-----  
\*\*ChatGPT:\*\*

好的，让我们开始编写一个更复杂的AI。首先，我们需要让AI能够创建所有的Terran单位和建筑。在Starcraft2中，每个单位和建筑都有一个特定的动作ID，我们需要使用这些ID来创建单位和建筑。

以下是一个基本的例子，展示了如何让AI创建一个SCV（Terran的基本工作单位）和一个Barracks（一个基本的军事建筑）：

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranBasicAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranBasicAgent, self).step(obs)


        # 如果我们没有scv, 就创建一个
        scvs = [unit for unit in obs.observation.feature_units
                 if unit.unit_type == units.Terran.SCV]
        if len(scvs) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_SCV_quick.id):
                return actions.FUNCTIONS.Train_SCV_quick("now")

        # 如果我们没有Barracks, 就创建一个
        barracks = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Barracks]
        if len(barracks) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Barracks_screen.id):
                return actions.FUNCTIONS.Build_Barracks_screen("now", (0, 0))

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们首先检查我们是否有SCV。如果没有，我们就创建一个。然后，我们检查我们是否有Barracks。如果没有，我们就创建一个。注意，我们需要使用can\_do函数来检查我们是否可以执行一个动作。例如，我们可能没有足够的资源来创建一个SCV，或者我们可能已经在创建一个SCV，所以我们不能再创建一个。

然后，我们需要让我们的单位能够施放所有可用的技能。这需要我们知道每个单位的技能和相应的动作ID。例如，Marine（一个基本的军事单位）有一个技能叫做"Stimpack"，我们可以使用以下代码来让Marine施放这个技能：

```
python  Copy code
marines = [unit for unit in obs.observation.feature_units
            if unit.unit_type == units.Terran.Marine]
if len(marines) > 0:
    marine = marines[0]
    if self.can_do(obs, actions.FUNCTIONS.Effect_Stim_quick.id):
        return actions.FUNCTIONS.Effect_Stim_quick("now", marine.tag)
```

在这个例子中，我们首先找到所有的Marine，然后选择第一个Marine，然后检查我们是否可以施放"Stimpack"技能。如果可以，我们就施放这个技能。


请注意，这只是一个基本的例子，实际的AI可能需要更复杂的逻辑来决定何时和如何创建单位和建筑，以及何时和如何施放技能。

-----  
\*\*You:\*\*

好的，那么完善补充所有的Terran单位和建筑到以上代码框架中

-----  
\*\*ChatGPT:\*\*

好的，让我们扩展上述代码以包括所有的Terran单位和建筑。请注意，这个例子仍然是非常基础的，它只是尝试创建每种类型的单位和建筑，而不是尝试执行任何特定的策略或战术。

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent
```

```

class TerranBasicAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranBasicAgent, self).step(obs)

        # 如果我们没有SCV, 就创建一个
        scvs = [unit for unit in obs.observation.feature_units
                 if unit.unit_type == units.Terran.SCV]
        if len(scvs) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_SCV_quick.id):
                return actions.FUNCTIONS.Train_SCV_quick("now")

        # 如果我们没有Barracks, 就创建一个
        barracks = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Barracks]
        if len(barracks) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Barracks_screen.id):
                return actions.FUNCTIONS.Build_Barracks_screen("now", (0, 0))

        # 如果我们没有Marine, 就创建一个
        marines = [unit for unit in obs.observation.feature_units
                   if unit.unit_type == units.Terran.Marine]
        if len(marines) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_Marine_quick.id):
                return actions.FUNCTIONS.Train_Marine_quick("now")

        # 如果我们没有Factory, 就创建一个
        factories = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Factory]
        if len(factories) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Factory_screen.id):
                return actions.FUNCTIONS.Build_Factory_screen("now", (0, 0))

        # 如果我们没有SiegeTank, 就创建一个
        siege_tanks = [unit for unit in obs.observation.feature_units
                       if unit.unit_type == units.Terran.SiegeTank]
        if len(siege_tanks) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_SiegeTank_quick.id):
                return actions.FUNCTIONS.Train_SiegeTank_quick("now")

        # 如果我们没有Starport, 就创建一个
        starports = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Starport]
        if len(starports) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Starport_screen.id):
                return actions.FUNCTIONS.Build_Starport_screen("now", (0, 0))

        # 如果我们没有Banshee, 就创建一个
        banshees = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Banshee]
        if len(banshees) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_Banshee_quick.id):
                return actions.FUNCTIONS.Train_Banshee_quick("now")

        return actions.FUNCTIONS.no_op()

```

这个AI将尝试创建一个SCV, 一个Barracks, 一个Marine, 一个Factory, 一个SiegeTank, 一个Starport和一个Banshee。请注意, 这个AI并没有考虑资源管理, 也没有考虑单位的位置或者战术。在实际的游戏中, 你可能需要更复杂的逻辑来决定何时和如何创建单位和建筑。

此外, 这个AI也没有处理单位的技能。每个单位都有自己的技能, 你可以根据游戏的情况来决定何时和如何使用这些技能。例如, 你可能需要让你的Marine使用Stimpack技能来增加他们的攻击速度, 或者让你的SiegeTank进入SiegeMode来增加他们的攻击范围。

最后, 这个AI也没有处理游戏的胜利条件。在Starcraft2中, 你的目标是摧毁敌人的所有建筑。你需要编写代码来让你的单位攻击敌人的建筑, 或者防御敌人的攻击。

这只是一个基础的AI, 你可以根据你的需要来扩展和改进它。如果你需要更多的帮助, 或者有任何问题, 随时告诉我!


-----  
 \*\*You:\*\*

把建筑和单位分开, 并且考虑相应的科技树解锁, 完善

-----  
 \*\*ChatGPT:\*\*

好的，让我们将建筑和单位的创建分开，并考虑科技树的解锁。在Starcraft2中，某些单位和建筑的创建需要先解锁相应的科技树。例如，要创建一个SiegeTank，你需要先有一个Factory和一个TechLab。

以下是一个更复杂的AI，它将建筑和单位的创建分开，并考虑科技树的解锁：

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranAdvancedAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranAdvancedAgent, self).step(obs)

        # 如果我们没有SCV，就创建一个
        scvs = [unit for unit in obs.observation.feature_units
                 if unit.unit_type == units.Terran.SCV]
        if len(scvs) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Train_SCV_quick.id):
                return actions.FUNCTIONS.Train_SCV_quick("now")

        # 如果我们没有Barracks，就创建一个
        barracks = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Barracks]
        if len(barracks) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Barracks_screen.id):
                return actions.FUNCTIONS.Build_Barracks_screen("now", (0, 0))

        # 如果我们没有Factory，就创建一个
        factories = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Factory]
        if len(factories) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Factory_screen.id):
                return actions.FUNCTIONS.Build_Factory_screen("now", (0, 0))

        # 如果我们没有Starport，就创建一个
        starports = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Starport]
        if len(starports) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Starport_screen.id):
                return actions.FUNCTIONS.Build_Starport_screen("now", (0, 0))

        # 如果我们没有Barracks，但没有Marine，就创建一个
        if len(barracks) > 0:
            marines = [unit for unit in obs.observation.feature_units
                      if unit.unit_type == units.Terran.Marine]
            if len(marines) == 0:
                if self.can_do(obs, actions.FUNCTIONS.Train_Marine_quick.id):
                    return actions.FUNCTIONS.Train_Marine_quick("now")

        # 如果我们没有Factory和TechLab，但没有SiegeTank，就创建一个
        techlabs = [unit for unit in obs.observation.feature_units
                   if unit.unit_type == units.Terran.FactoryTechLab]
        if len(factories) > 0 and len(techlabs) > 0:
            siege_tanks = [unit for unit in obs.observation.feature_units
                          if unit.unit_type == units.Terran.SiegeTank]
            if len(siege_tanks) == 0:
                if self.can_do(obs, actions.FUNCTIONS.Train_SiegeTank_quick.id):
                    return actions.FUNCTIONS.Train_SiegeTank_quick("now")

        # 如果我们没有Starport，但没有Banshee，就创建一个
        if len(starports) > 0:
            banshees = [unit for unit in obs.observation.feature_units
                      if unit.unit_type == units.Terran.Banshee]
            if len(banshees) == 0:
                if self.can_do(obs, actions.FUNCTIONS.Train_Banshee_quick.id):
                    return actions.FUNCTIONS.Train_Banshee_quick("now")

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们首先尝试创建所有的建筑，然后根据我们拥有的建筑来创建单位。例如，我们只有在我们有一个Barracks的时候才会尝试创建一个Marine，只有在我们有一个Factory和一个TechLab的时候才会尝试创建一个SiegeTank。

请注意，这个AI仍然是非常基础的，它并没有考虑资源管理，也没有考虑单位的位置或者战术。在实际的游戏中，你可能需要更复杂的逻辑来决定何时和如何创建单位和建筑。

此外，这个AI也没有处理单位的技能。每个单位都有自己的技能，你需要根据游戏的情况来决定何时和如何使用这些技能。例如，你可能需要让你的Marine使用Stimpack技能来增加他们的攻击速度，或者让你的SiegeTank进入SiegeMode来增加他们的攻击范围。

最后，这个AI也没有处理游戏的胜利条件。在Starcraft2中，你的目标是摧毁敌人的所有建筑。你需要编写代码来让你的单位攻击敌人的建筑，或者防御敌人的攻击。

这只是一个基础的AI，你可以根据你的需要来扩展和改进它。如果你需要更多的帮助，或者有任何问题，随时告诉我！

-----  
\*\*You:\*\*

加入资源管理部分，考虑训练单位的分配和训练时间

-----  
\*\*ChatGPT:\*\*

在Starcraft2中，资源管理是非常重要的。你需要确保你有足够的资源来创建单位和建筑，同时也需要确保你的资源不会被浪费。此外，你还需要考虑训练单位的时间。如果你的训练队列太长，你可能会浪费资源，因为你的建筑在等待训练队列的时候不能做其他的事情。

以下是一个考虑资源管理和训练时间的AI：

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranResourceManagementAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranResourceManagementAgent, self).step(obs)

        # 如果我们没有SCV，就创建一个
        scvs = [unit for unit in obs.observation.feature_units
                 if unit.unit_type == units.Terran.SCV]
        if len(scvs) < 16: # 限制SCV的数量，以避免浪费资源
            if self.can_do(obs, actions.FUNCTIONS.Train_SCV_quick.id):
                return actions.FUNCTIONS.Train_SCV_quick("now")

        # 如果我们没有Barracks，就创建一个
        barracks = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Barracks]
        if len(barracks) < 3: # 限制Barracks的数量，以避免浪费资源
            if self.can_do(obs, actions.FUNCTIONS.Build_Barracks_screen.id):
                return actions.FUNCTIONS.Build_Barracks_screen("now", (0, 0))

        # 如果我们有Barracks，但没有Marine，就创建一个
        if len(barracks) > 0:
            marines = [unit for unit in obs.observation.feature_units
                        if unit.unit_type == units.Terran.Marine]
            if len(marines) < 3 * len(barracks): # 每个Barracks最多可以同时训练3个Marine
                if self.can_do(obs, actions.FUNCTIONS.Train_Marine_quick.id):
                    return actions.FUNCTIONS.Train_Marine_quick("now")

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们限制了SCV和Barracks的数量，以避免浪费资源。我们还限制了每个Barracks可以同时训练的Marine的数量，以避免浪费训练时间。

请注意，这只是一个基础的例子，实际的AI可能需要更复杂的逻辑来管理资源和训练时间。例如，你可能需要考虑你的当前资源和预期的资源收入，以决定你应该创建哪种单位和建筑。你也可能需要考虑你的当前军事力量和预期的军事需求，以决定你应该训练哪种单位。

如果你需要更多的帮助，或者有任何问题，随时告诉我！

-----  
\*\*You:\*\*

将各部分相结合为一个完整的项目，并提出还需要补充哪些内容

-----  
 \*\*ChatGPT:\*\*

好的，让我们将所有的部分结合起来，创建一个更完整的Starcraft2 AI。这个AI将尝试创建和管理单位和建筑，同时也会考虑资源管理和训练时间。

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranCompleteAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranCompleteAgent, self).step(obs)

        # 如果我们没有scv, 就创建一个
        scvs = [unit for unit in obs.observation.feature_units
                 if unit.unit_type == units.Terran.SCV]
        if len(scvs) < 16: # 限制SCV的数量, 以避免浪费资源
            if self.can_do(obs, actions.FUNCTIONS.Train_SCV_quick.id):
                return actions.FUNCTIONS.Train_SCV_quick("now")

        # 如果我们没有Barracks, 就创建一个
        barracks = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Barracks]
        if len(barracks) < 3: # 限制Barracks的数量, 以避免浪费资源
            if self.can_do(obs, actions.FUNCTIONS.Build_Barracks_screen.id):
                return actions.FUNCTIONS.Build_Barracks_screen("now", (0, 0))

        # 如果我们有Barracks, 但没有Marine, 就创建一个
        if len(barracks) > 0:
            marines = [unit for unit in obs.observation.feature_units
                       if unit.unit_type == units.Terran.Marine]
            if len(marines) < 3 * len(barracks): # 每个Barracks最多可以同时训练3个Marine
                if self.can_do(obs, actions.FUNCTIONS.Train_Marine_quick.id):
                    return actions.FUNCTIONS.Train_Marine_quick("now")

        # 如果我们没有Factory, 就创建一个
        factories = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Factory]
        if len(factories) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Factory_screen.id):
                return actions.FUNCTIONS.Build_Factory_screen("now", (0, 0))

        # 如果我们有Factory和TechLab, 但没有SiegeTank, 就创建一个
        techlabs = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.FactoryTechLab]
        if len(factories) > 0 and len(techlabs) > 0:
            siege_tanks = [unit for unit in obs.observation.feature_units
                           if unit.unit_type == units.Terran.SiegeTank]
            if len(siege_tanks) == 0:
                if self.can_do(obs, actions.FUNCTIONS.Train_SiegeTank_quick.id):
                    return actions.FUNCTIONS.Train_SiegeTank_quick("now")

        # 如果我们没有Starport, 就创建一个
        starports = [unit for unit in obs.observation.feature_units
                     if unit.unit_type == units.Terran.Starport]
        if len(starports) == 0:
            if self.can_do(obs, actions.FUNCTIONS.Build_Starport_screen.id):
                return actions.FUNCTIONS.Build_Starport_screen("now", (0, 0))

        # 如果我们有Starport, 但没有Banshee, 就创建一个
        if len(starports) > 0:
            banshees = [unit for unit in obs.observation.feature_units
                        if unit.unit_type == units.Terran.Banshee]
            if len(banshees) == 0:
                if self.can_do(obs, actions.FUNCTIONS.Train_Banshee_quick.id):
                    return actions.FUNCTIONS.Train_Banshee_quick("now")

        return actions.FUNCTIONS.no_op()
```

这个AI现在可以创建和管理单位和建筑，并且会考虑资源管理和训练时间。然而，这个AI仍然非常基础，还有很多可以改进的地方。以下是一些可能的改进：

1. 单位的位置和战术：这个AI并没有考虑单位的位置或者战术。例如，你可能需要让你的单位在特定的位置防守，或者让你的单位攻击敌人的特定建筑。
2. 单位的技能：这个AI并没有处理单位的技能。每个单位都有自己的技能，你需要根据游戏的情况来决定何时和如何使用这些技能。
3. 游戏的胜利条件：这个AI并没有处理游戏的胜利条件。在Starcraft2中，你的目标是摧毁敌人的所有建筑。你需要编写代码来让你的单位攻击敌人的建筑，或者防御敌人的攻击。
4. 更复杂的资源管理：这个AI的资源管理非常基础。在实际的游戏中，你可能需要更复杂的逻辑来管理你的资源。例如，你可能需要考虑你的当前资源和预期的资源收入，以决定你应该创建哪种单位和建筑。
5. 更复杂的训练时间管理：这个AI的训练时间管理也非常基础。在实际的游戏中，你可能需要更复杂的逻辑来管理你的训练时间。例如，你可能需要考虑你的当前军事力量和预期的军事需求，以决定你应该训练哪种单位。

如果你需要更多的帮助，或者有任何问题，随时告诉我！

-----  
\*\*You:\*\*

接下来补充处理单位的技能部分。根据游戏的情况来决定每个单位何时和如何使用各自的技能。

-----  
\*\*ChatGPT:\*\*

好的，让我们添加一些代码来处理单位的技能。在Starcraft2中，每个单位都有自己的技能，你需要根据游戏的情况来决定何时和如何使用这些技能。

例如，让我们添加一些代码来让Marine使用他们的Stimpack技能，让SiegeTank进入SiegeMode，让Banshee使用他们的Cloak技能：

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranSkillfulAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranSkillfulAgent, self).step(obs)

        # 如果我们有Marine，就让他们使用Stimpack技能
        marines = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Marine]
        for marine in marines:
            if self.can_do(obs, actions.FUNCTIONS.Effect_Stim_quick.id):
                return actions.FUNCTIONS.Effect_Stim_quick("now", marine.tag)

        # 如果我们有SiegeTank，就让他们进入SiegeMode
        siege_tanks = [unit for unit in obs.observation.feature_units
                       if unit.unit_type == units.Terran.SiegeTank]
        for siege_tank in siege_tanks:
            if self.can_do(obs, actions.FUNCTIONS.Morph_SiegeMode_quick.id):
                return actions.FUNCTIONS.Morph_SiegeMode_quick("now", siege_tank.tag)

        # 如果我们有Banshee，就让他们使用Cloak技能
        banshees = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Banshee]
        for banshee in banshees:
            if self.can_do(obs, actions.FUNCTIONS.Behavior_CloakOn_quick.id):
                return actions.FUNCTIONS.Behavior_CloakOn_quick("now", banshee.tag)

        # 在这里添加你的其他代码...

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们首先找到所有的Marine，然后让他们使用Stimpack技能。然后，我们找到所有的SiegeTank，然后让他们进入SiegeMode。最后，我们找到所有的Banshee，然后让他们使用Cloak技能。

请注意，这只是一个基础的例子，实际的AI可能需要更复杂的逻辑来决定何时和如何使用技能。例如，你可能需要考虑你的单位的生命值，敌人的位置，或者其他的游戏情况。你也可能需要考虑你的资源，因为一些技能需要消耗资源。

如果你需要更多的帮助，或者有任何问题，随时告诉我！

**\*\*You:\*\***

请接着补充所有Terran单位技能管理到TerranSkillfulAgent

**\*\*ChatGPT:\*\***

好的，让我们添加更多的代码来处理所有Terran单位的技能。请注意，这个例子仍然是非常基础的，它只是尝试使用每种类型的技能，而不是尝试执行任何特定的策略或战术。

```
python  Copy code
from pysc2.lib import actions, features, units
from pysc2.agents import base_agent

class TerranSkillfulAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranSkillfulAgent, self).step(obs)

        # 如果我们有Marine, 就让他们使用Stimpack技能
        marines = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Marine]
        for marine in marines:
            if self.can_do(obs, actions.FUNCTIONS.Effect_Stim_quick.id):
                return actions.FUNCTIONS.Effect_Stim_quick("now", marine.tag)

        # 如果我们有SiegeTank, 就让他们进入SiegeMode
        siege_tanks = [unit for unit in obs.observation.feature_units
                       if unit.unit_type == units.Terran.SiegeTank]
        for siege_tank in siege_tanks:
            if self.can_do(obs, actions.FUNCTIONS.Morph_SiegeMode_quick.id):
                return actions.FUNCTIONS.Morph_SiegeMode_quick("now", siege_tank.tag)

        # 如果我们有Banshee, 就让他们使用Cloak技能
        banshees = [unit for unit in obs.observation.feature_units
                    if unit.unit_type == units.Terran.Banshee]
        for banshee in banshees:
            if self.can_do(obs, actions.FUNCTIONS.Behavior_CloakOn_quick.id):
                return actions.FUNCTIONS.Behavior_CloakOn_quick("now", banshee.tag)

        # 如果我们有Raven, 就让他们使用AutoTurret技能
        ravens = [unit for unit in obs.observation.feature_units
                  if unit.unit_type == units.Terran.Raven]
        for raven in ravens:
            if self.can_do(obs, actions.FUNCTIONS.Effect_AutoTurret_screen.id):
                return actions.FUNCTIONS.Effect_AutoTurret_screen("now", (0, 0))

        # 如果我们有Battlecruiser, 就让他们使用YamatoGun技能
        battlecruisers = [unit for unit in obs.observation.feature_units
                          if unit.unit_type == units.Terran.Battlecruiser]
        for battlecruiser in battlecruisers:
            if self.can_do(obs, actions.FUNCTIONS.Effect_YamatoGun_screen.id):
                return actions.FUNCTIONS.Effect_YamatoGun_screen("now", (0, 0))

        # 在这里添加你的其他代码...

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们添加了代码来让Raven使用AutoTurret技能，让Battlecruiser使用YamatoGun技能。请注意，这只是一个基础的例子，实际的AI可能需要更复杂的逻辑来决定何时和如何使用技能。例如，你可能需要考虑你的单位的生命值，敌人的位置，或者其他的游戏情况。你也可能需要考虑你的资源，因为一些技能需要消耗资源。

这个AI现在可以处理一些基本的单位技能，但是还有很多可以改进的地方。以下是一些可能的改进：

1. 更复杂的技能管理：这个AI的技能管理非常基础。在实际的游戏中，你可能需要更复杂的逻辑来管理你的技能。例如，你可能需要考虑你的单位的生命值，敌人的位置，或者其他的游戏情况。
2. 更复杂的资源管理：这个AI的资源管理非常基础。在实际的游戏中，你可能需要更复杂的逻辑来管理你的资源。例如，你可能需要考虑你的当前资源和预期的资源收入，以决定你应该使用哪种技能。
3. 更复杂的训练时间管理：这个AI的训练时间管理也非常基础。在实际的游戏中，你可能需要更复杂的逻辑来管理你的训练时间。例如，你可能需要考虑你的当前军事力量和预期的军事需求，以决定你应该训练哪种单位。



4. 游戏的胜利条件：这个AI并没有处理游戏的胜利条件。在Starcraft2中，你的目标是摧毁敌人的所有建筑。你需要编写代码来让你的单位攻击敌人的建筑，或者防御敌人的攻击。

如果你需要更多的帮助，或者有任何问题，随时告诉我！

-----  
\*\*You:\*\*

向TerranSkillfulAgent中添加Battlecruiser的折跃技能，在战斗中保存实力很重要，因此单位的技能不能随便施放，你还需要计算合适的施放各自技能的情况和时机。

-----  
\*\*ChatGPT:\*\*

你完全正确，单位的技能不能随便施放，需要根据战斗的情况和时机来决定。例如，Battlecruiser的折跃技能可以让它瞬间移动到地图上的任何位置，这个技能可以用来突然发动攻击，或者从战斗中撤退。

让我们添加一些代码来让Battlecruiser在生命值低于一定阈值时使用折跃技能：

```
python  Copy code
from pyc2.lib import actions, features, units
from pyc2.agents import base_agent

class TerranSkillfulAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranSkillfulAgent, self).step(obs)

        # 如果我们有Battlecruiser，就让他们使用YamatoGun技能，或者在生命值低于一定阈值时使用折跃技能
        battlecruisers = [unit for unit in obs.observation.feature_units
                           if unit.unit_type == units.Terran.Battlecruiser]
        for battlecruiser in battlecruisers:
            if battlecruiser.health / battlecruiser.health_max < 0.5: # 如果生命值低于50%
                if self.can_do(obs, actions.FUNCTIONS.Effect_TacticalJump_minimap.id):
                    return actions.FUNCTIONS.Effect_TacticalJump_minimap("now", (0, 0)) # 折跃到地图的左上角
            else:
                if self.can_do(obs, actions.FUNCTIONS.Effect_YamatoGun_screen.id):
                    return actions.FUNCTIONS.Effect_YamatoGun_screen("now", (0, 0))

        # 在这里添加你的其他代码...

        return actions.FUNCTIONS.no_op()
```

在这个例子中，我们让Battlecruiser在生命值低于50%时使用折跃技能，否则使用YamatoGun技能。请注意，我们需要指定折跃的目标位置，这个位置应该根据游戏的情况来决定。在这个例子中，我们简单地让Battlecruiser折跃到地图的左上角。

请注意，这只是一个基础的例子，实际的AI可能需要更复杂的逻辑来决定何时和如何使用技能。例如，你可能需要考虑你的单位的生命值，敌人的位置，或者其他的游戏情况。你也可能需要考虑你的资源，因为一些技能需要消耗资源。

如果你需要更多的帮助，或者有任何问题，随时告诉我！


-----  
\*\*You:\*\*

Good，以此逻辑跟进完善TerranSkillfulAgent其它所有Terran单位的技能，并考虑到集中火力攻击enemy或其它火力攻击方式，即计算最优火力方案

-----  
\*\*ChatGPT:\*\*

在Starcraft2中，火力集中是一个重要的战术。你需要让你的单位集中火力攻击敌人的某个单位，以尽快消灭它。同时，你也需要考虑你的单位的位置和敌人的位置，以决定最优的火力方案。

以下是一个更复杂的AI，它会尝试让所有的单位集中火力攻击敌人的某个单位，并会根据单位的生命值和技能来决定何时使用技能：

```
python  Copy code
from pyc2.lib import actions, features, units
from pyc2.agents import base_agent
```

```

class TerranSkillfulAgent(base_agent.BaseAgent):
    def step(self, obs):
        super(TerranSkillfulAgent, self).step(obs)

        # 找到敌人的单位
        enemies = [unit for unit in obs.observation.feature_units
                    if unit.alliance == features.PlayerRelative.ENEMY]

        # 如果有敌人, 就让所有的单位集中火力攻击敌人的第一个单位
        if len(enemies) > 0:
            target = enemies[0].tag

            # 如果我们有Marine, 就让他们使用Stimpack技能, 然后攻击敌人
            marines = [unit for unit in obs.observation.feature_units
                        if unit.unit_type == units.Terran.Marine]
            for marine in marines:
                if self.can_do(obs, actions.FUNCTIONS.Effect_Stim_quick.id):
                    return actions.FUNCTIONS.Effect_Stim_quick("now", marine.tag)
                if self.can_do(obs, actions.FUNCTIONS.Attack_Attack_screen.id):
                    return actions.FUNCTIONS.Attack_Attack_screen("now", enemies[0].position)

            # 如果我们有SiegeTank, 就让他们进入SiegeMode, 然后攻击敌人
            siege_tanks = [unit for unit in obs.observation.feature_units
                            if unit.unit_type == units.Terran.SiegeTank]
            for siege_tank in siege_tanks:
                if self.can_do(obs, actions.FUNCTIONS.Morph_SiegeMode_quick.id):
                    return actions.FUNCTIONS.Morph_SiegeMode_quick("now", siege_tank.tag)
                if self.can_do(obs, actions.FUNCTIONS.Attack_Attack_screen.id):
                    return actions.FUNCTIONS.Attack_Attack_screen("now", enemies[0].position)

            # 如果我们有Banshee, 就让他们使用Cloak技能, 然后攻击敌人
            banshees = [unit for unit in obs.observation.feature_units
                         if unit.unit_type == units.Terran.Banshee]
            for banshee in banshees:
                if self.can_do(obs, actions.FUNCTIONS.Behavior_CloakOn_quick.id):
                    return actions.FUNCTIONS.Behavior_CloakOn_quick("now", banshee.tag)
                if self.can_do(obs, actions.FUNCTIONS.Attack_Attack_screen.id):
                    return actions.FUNCTIONS.Attack_Attack_screen("now", enemies[0].position)

            # 如果我们有Raven, 就让他们使用AutoTurret技能, 然后攻击敌人
            ravens = [unit for unit in obs.observation.feature_units
                       if unit.unit_type == units.Terran.Raven]
            for raven in ravens:
                if self.can_do(obs, actions.FUNCTIONS.Effect_AutoTurret_screen.id):
                    return actions.FUNCTIONS.Effect_AutoTurret_screen("now", enemies[0].position)
                if self.can_do(obs, actions.FUNCTIONS.Attack_Attack_screen.id):
                    return actions.FUNCTIONS.Attack_Attack_screen("now", enemies[0].position)

            # 如果我们有Battlecruiser, 就让他们使用YamatoGun技能, 或者在生命值低于一定阈值时使用折跃技能
            battlecruisers = [unit for unit in obs.observation.feature_units
                               if unit.unit_type == units.Terran.Battlecruiser]
            for battlecruiser in battlecruisers:
                if battlecruiser.health / battlecruiser.health_max < 0.5: # 如果生命值低于50%
                    if self.can_do(obs, actions.FUNCTIONS.Effect_TacticalJump_minimap.id):
                        return actions.FUNCTIONS.Effect_TacticalJump_minimap("now", (0, 0)) # 折跃到地图的左上角
                else:
                    if self.can_do(obs, actions.FUNCTIONS.Effect_YamatoGun_screen.id):
                        return actions.FUNCTIONS.Effect_YamatoGun_screen("now", enemies[0].position)
                    if self.can_do(obs, actions.FUNCTIONS.Attack_Attack_screen.id):
                        return actions.FUNCTIONS.Attack_Attack_screen("now", enemies[0].position)

            # 在这里添加你的其他代码...

        return actions.FUNCTIONS.no_op()

```

在这个例子中, 我们首先找到所有的敌人, 然后让所有的单位集中火力攻击敌人的第一个单位。我们还根据单位的生命值和技能来决定何时使用技能。

请注意, 这只是一个基础的例子, 实际的AI可能需要更复杂的逻辑来决定何时和如何使用技能和攻击敌人。例如, 你可能需要考虑你的单位的生命值, 敌人的位置, 或者其他的游戏情况。你也可能需要考虑你的资源, 因为一些技能需要消耗资源。

如果你需要更多的帮助, 或者有任何问题, 随时告诉我!